Deep Reinforcement Learning Assisted Automated Guiding Vector Selection for Large-scale Sparse Multi-objective Optimization

Article *in* Swarm and Evolutionary Computation · May 2024

CITATIONS
0

READS
93

3 authors:

Shuai Shao
Anhui University
9 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Ye Tian
Anhui University
130 PUBLICATIONS **8,426** CITATIONS
SEE PROFILE

Xingyi Zhang
Anhui University
238 PUBLICATIONS **11,074** CITATIONS
SEE PROFILE

Deep Reinforcement Learning Assisted Automated Guiding Vector Selection for Large-scale Sparse Multi-objective Optimization

Shuai Shao[a], Ye Tian[a,*] and Xingyi Zhang[a]

[a]School of Computer Science and Technology, Anhui University, Hefei, 230601, China

ARTICLE INFO

Keywords:
Multi-objective optimization
Sparse optimization
Evolutionary computation
Deep reinforcement learning
Variable clustering

ABSTRACT

Sparse multi-objective optimization problems (SMOPs) are prevalent in a wide range of applications, spanning from the fields of science to engineering. Existing sparse evolutionary algorithms utilize single or multiple guiding vectors to direct the generation of offspring solutions in a constant manner according to human experience, which are difficult to determine the best guiding vector for various population states and prone to falling into premature convergence. To address the dilemma in guiding vector adaptation, this paper proposes a novel guiding vector selection method based on reinforcement learning. In the proposed method, the features extracted from the current population are regarded as states, the overall degrees of improvement in population convergence and diversity are regarded as rewards, and the candidate guiding vectors are regarded as actions. By using deep neural networks to establish the mapping model between population states and the expected cumulative rewards of guiding vectors, the proposed method can determine the best guiding vector for the current population at each generation. The selected guiding vector inspires the development of novel genetic operators, which can approximate sparse Pareto optimal solutions in high-dimensional decision spaces. Experimental results on both benchmark and real-world SMOPs demonstrate that the proposed algorithm has significant advantages when compared with the state-of-the-art.

1. Introduction

Sparse multi-objective optimization problems (SMOPs) refer to problems with sparse optimal solutions, i.e., most variables in the optimal solutions are zero, which widely exist in real-world applications. For example, the critical node detection problem [23, 31] is to minimize the pairwise connectivity and the number of deleted nodes, the feature selection problem [30, 20] is to minimize the error of the model and the number of selected features, the location selection of distribution centers problem [45, 46] is to minimize the workload and the transmission risk, and the instance selection problem [7, 6] is to minimize the error of the model and the number of selected instances. Since most SMOPs rely on large-scale datasets in science and engineering domains, they involve a large number of decision variables, called large-scale SMOPs.

During the past decade, a number of multi-objective evolutionary algorithms (MOEAs) for solving large-scale multi-objective optimization problems (MOPs) have been developed [42, 13, 32, 17, 35], which can be roughly divided into three categories. The first category uses divide-and-conquer strategies [28] to divide decision variables into several groups, and then optimizes each group of decision variables alternately. The second category adopts transformation strategies [14] to transform the original problems into low-dimensional sub-problems, hence reducing the dimensionality of the decision variable. The last category employs novel search strategies [27] to effectively generate offspring solutions. In spite of the effectiveness of these strategies on large-scale MOPs, they cannot provide good optimization performance for large-scale SMOPs, as they do not consider the sparse characteristic of SMOPs' Pareto optimal solutions. In order to deal with large-scale SMOPs, some algorithms design new genetic operators to efficiently identify zero variables. To be specific, these algorithms optimize a binary vector for dimensionality reduction and a real vector for finding the optimal values of nonzero variables. When binary vectors are sufficiently optimized, searching for real vectors becomes a small-scale optimization problem that can be easily handled with general strategies [33, 8, 9]. Therefore, the challenge in solving large-scale SMOPs lies in effectively determining nonzero positions in a binary vector.

Existing MOEAs customized for large-scale SMOPs mostly identify zero variables in binary vectors through one or multiple guiding vectors, where the dimensionality of each guiding vector equals that of the decision variables, and each element of the guiding vector represents the probability of the corresponding decision variable being zero. Due to the sparsity of SMOPs taken into consideration, the guiding vector can direct genetic operators to rapidly approximate the sparse Pareto front. For example, SparseEA [44] and SparseEA2 [48] perform precalculation of a guiding vector to assess the significance of each decision variable, and flip binary variables according to the values of the guiding vector. Since the guiding vector is constant during the evolutionary process, MGCEA [41] suggests a dynamic guiding vector based on the sparse distribution of the current population and groups decision variables that have a similar value in the guiding vector. To maintain the diversity of the solutions, MP-MMEA [38] and HHC-MMEA [11] use different guiding vectors to evolve multiple subpopulations, which can approximate multiple sparse Pareto optimal solution sets. To enhance the ability to explore decision spaces,

*Corresponding author.
freshshao@gmail.com (S. Shao); field910921@gmail.com (Y. Tian); xyzhanghust@gmail.com (X. Zhang)
ORCID(s):

DSGEA [50] combines a static and a dynamic guide vector to cluster decision variables, so that the decision variable grouping can be more reasonable.

Guiding vectors can direct evolutionary algorithms to approximate sparse Pareto optimal solutions, which are crucial for solving large-scale SMOPs effectively. Compared with MOEAs using a single guiding vector, MOEAs using multiple guiding vectors can exhibit various search behaviors for better versatility. However, existing MOEAs determine the opportunity and scope of utilizing a guiding vector in constant manners, which are prone to falling into local optima and waste many function evaluations. Deep reinforcement learning has a strong capability to handle complex environments, adapting to different scenarios and making corresponding decisions. Therefore, to reduce the probability of getting trapped into local optima, this paper aims to address this dilemma via reinforcement learning, where the main contributions are reflected in the following three aspects:

1. A deep reinforcement learning assisted guiding vector selection method is proposed. By regarding the features extracted from the current population as states, the overall degrees of improvement in population convergence and diversity as rewards, and the candidate guiding vectors as actions, an agent uses deep neural networks to establish the mapping model between the population states and the guiding vectors. The model can automatically determine the best guiding vector according to the current population state to maximize the cumulative performance improvement during the evolutionary process. To the best of the authors' knowledge, this work serves as the first attempt to use reinforcement learning to adaptively select guiding vectors.
2. An adaptive variable clustering method is proposed, which categorizes all decision variables into two groups of varying sizes according to the selected guiding vector and the sparsity of the current population, further inspiring the development of a crossover operator and a mutation operator. The crossover operator flips all binary variables in a selected group, which can globally perform dimensionality reduction when selecting the group with a larger size and fine-tune the sparse population when selecting the group with a smaller size. The mutation operator can adaptively adjust the mutation probability based on the diversity of the strategies recommended by reinforcement learning, which can reduce the risk of falling into local optima.
3. An automated guiding vector selection-based evolutionary algorithm, abbreviated as AGSEA, is developed based on the proposed guiding vector selection method, adaptive clustering method, crossover operator, and mutation operator. In the proposed algorithm, the agent periodically updates the weights of deep neural networks to direct the selection of guiding vectors. By adopting three different guiding vectors as candidate actions, the proposed algorithm does not easily fall into local optima. Compared with the state-of-the-art evolutionary algorithms, the proposed algorithm shows effectiveness and superiority in solving both benchmark and real-world SMOPs.

The structure of the remainder is organized as follows. In Section 2, we briefly review fundamental concepts related to SMOPs, existing sparse MOEAs and automated selection methods, then raise the motivation of this work. In Section 3, a detailed explanation of the proposed MOEA is presented. Experimental results and analysis on benchmark and real-world SMOPs are given in Section 4. Finally, conclusions and future work are drawn in Section 5.

2. Related works

2.1. Large-scale SMOPs

Generally, an unconstrained MOP can be formulated as

$$\begin{aligned} &\text{Minimize } f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})) \\ &\text{Subject to } \mathbf{x} = (x_1, x_2, \dots, x_d) \in \Omega \end{aligned}, \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \dots, x_d)$ denotes an d-dimensional decision variable vector in the decision space Ω, and $f(\mathbf{x})$ denotes its objective vector containing m objective values [43, 47, 34]. If solution $\mathbf{x_1}$ and solution $\mathbf{x_2}$ satisfy $\forall i : f_i(\mathbf{x_1}) \leq f_i(\mathbf{x_2}) \wedge \exists j : f_j(\mathbf{x_1}) < f_j(\mathbf{x_2})$, where $i, j \in \{1, 2, \dots, m\}$, it is said that $\mathbf{x}$ dominates $\mathbf{y}$ (denoted as $f(\mathbf{x_1}) \prec f(\mathbf{x_2})$). If there exists no solution that dominates $\mathbf{x}^*$, then solution $\mathbf{x}^*$ is a Pareto optimal solution and all the Pareto optimal solutions compose the Pareto set.

If an MOP contains a great number of decision variables and most decision variables should be optimized to zero, the optimization problem is called a large-scale SMOP [44]. The aim of solving a large-scale SMOP is to find the sparse Pareto optimal solutions in a large decision space. The performance of traditional MOEAs often deteriorates dramatically when solving large-scale SMOPs due to the high dimensional search space [5] and sparse nature of optimal solutions. Therefore, it is necessary to generate sparse solutions efficiently and accurately for solving large-scale SMOPs. To achieve this goal, some tailored search strategies have been proposed recently, the evolutionary algorithms incorporating which are known as sparse MOEAs and are introduced in the next subsection.

2.2. Existing MOEAs for large-scale SMOPs

Since the concept of sparse multi-objective optimization was first proposed in 2020 [44], a number of MOEAs have been designed for SMOPs. In [44], a bi-level encoding scheme consisting of a binary vector and a real vector is used, where the binary vector is optimized to search for the optimal sparse distribution and the real vector is optimized by simulated binary crossover [8] and polynomial mutation [9]. The bi-level encoding scheme is now used by most MOEAs for solving large-scale SMOPs. To have a better understanding of how these MOEAs work, in this subsection,

we review some representative MOEAs for SMOPs, which can be roughly divided into three categories.

The first category covers new variation operator-based sparse MOEAs. To enhance the search efficiency, the most intuitive idea is to directly indicate the probability of being zero for each decision variable and guide the generation of new solutions accordingly. For example, SparseEA [44] and SparseEA2 [48] calculate a guiding vector determining the critical degree of each decision variable separately before the iteration process, which inspires the flip of the binary vector. More specifically, non-critical variables can be easily flipped to zero and vice versa. To synchronize the changes of binary vectors and real vectors, TS-SparseEA [19] proposes a matching approach that can enable the binary vector of each solution to match with a real vector in the current solutions based on the similarity of their mutations.

The second category is known as the dimensionality reduction-based sparse MOEAs. MOEA/PSL [40] employs unsupervised neural networks to learn the Pareto optimal subspace during the evolutionary process. At each generation, the non-dominated solutions are used to train a restricted Boltzmann machine and a denoising autoencoder, aiming to learn the sparse distribution and the compact representation of decision variables, respectively. PM-MOEA [39] proposes an evolutionary pattern mining algorithm to extract non-zero variables from the current solutions. In PM-MOEA, the Pareto optimal solutions are treated as transaction datasets, and an evolutionary pattern mining method is proposed to extract the maximum and minimum candidate sets. Then, when generating offspring solutions, the variables in the candidate sets are determined by the proposed genetic operators, and the remaining variables are fixed at zero, which can effectively limit the dimensionality of the generated solutions.

The third category of sparse MOEAs reduces the number of decision variables by grouping variables, where the idea is to divide decision variables into a number of groups and flip all variables within a group simultaneously. To more efficiently search in the decision space with millions of dimensions, SLMEA [36] proposes a fast clustering method to divide all the decision variables into multiple groups, where a single variable is used to represent all variables within the same group, greatly reducing the number of decision variables. DSGEA [50] combines a static and dynamic guiding vector to calculate scores for each decision variable. Then, based on the distribution of non-zero decision variables in the current population, similar decision variables are grouped together. In DSGEA, the decision variables in each group are flipped according to their scores, which enhances the detection of non-zero variables and improves the search efficiency. Since each group is randomly selected when generating offspring solutions, MGCEA [41] divides all decision variables into multiple groups with different probabilities of being zero, hence the decision space can be more effectively explored and exploited.

For sparse MOEAs based on dimensionality reduction and variable clustering, they are prone to falling into local optima by optimizing shortened decision vectors. While for sparse MOEAs based on new variation operators, they use a single or multiple guiding vectors to direct the generation of offspring solutions, which possesses better exploration capabilities. Fig. 1 shows the convergence profiles of three variants of a new variation operator-based algorithm MSKEA [12] using different single guiding vectors. It can be observed that for different SMOPs, the corresponding best guiding vector varies accordingly, so it is crucial to select a suitable guiding vector for solving various large-scale SMOPs. However, existing MOEAs rely on human experience to determine the opportunity and scope for employing different guiding vectors, which leads to considerable subjectivity.

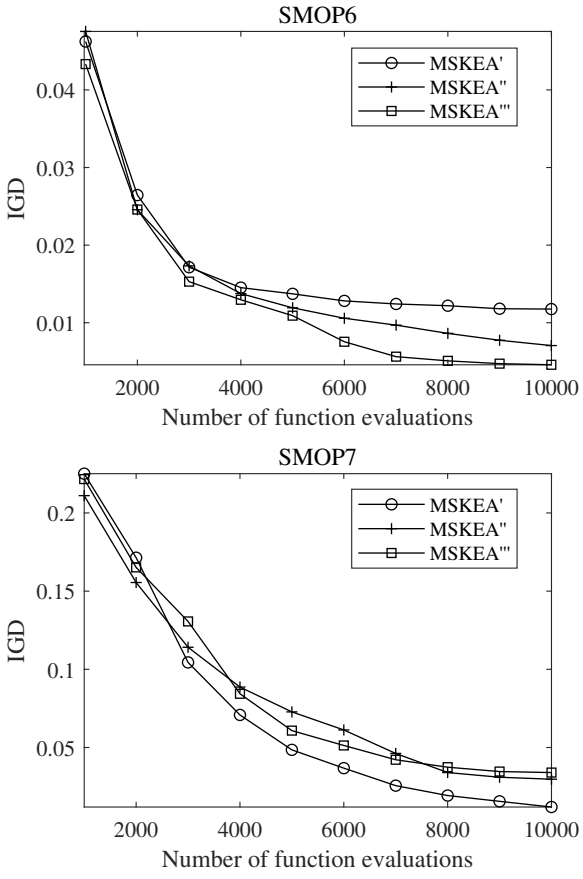

Figure 1: Convergence profiles of IGD values obtained by MSKEA′, MSKEA″, MSKEA‴ on SMOP6 and SMOP7.

2.3. Automated strategy selection methods

In order to improve the performance of MOEAs, some work has been dedicated to achieving automated strategy selection based on deep reinforcement learning during the evolutionary process. In [25], an end-to-end framework for solving multiobjective travelling salesman problems (MOTSPs) is proposed, which decomposes the MOTSP into a set of subproblems and models each subproblem in a neural network manner. The neural network can be adapted to learn how to select the next city given the city information and the selected cities by deep reinforcement learning

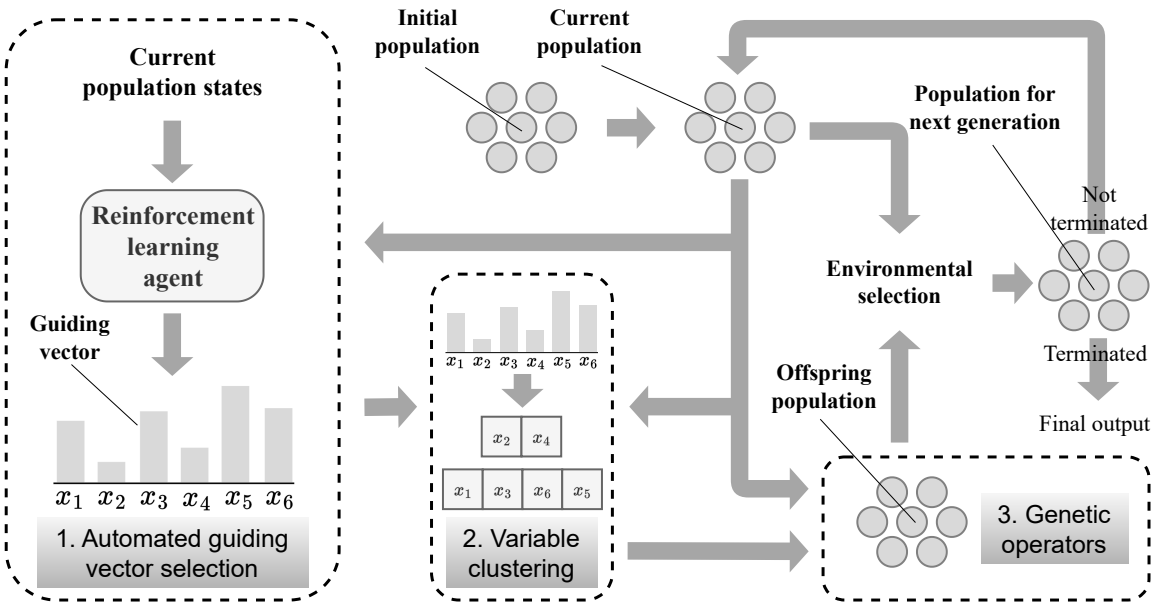

Figure 2: General procedure of the proposed AGSEA includes three core components: 1. automated guiding vector selection, 2. adaptive variable clustering, and 3. novel genetic operators. These components are enclosed in bold dashed boxes.

methods. To select promising operators for different optimization problems, MOEA/D-DQN [37] establishes a deep reinforcement learning-based mapping model between the decision variables of each solution in the current population and the candidate operators, which can achieve superior performance on different types of MOPs. Since MOEA/D-DQN does not extract features from the population, PKAEO [51] adopts the convergence and diversity of the current population in the decision spaces as the population states, which can characterize populations more effectively, thus establishing a more accurate mapping model between population states and candidate operators. In [15], MOEA/DRL uses deep reinforcement learning networks to mine sparse nonzero variables for reducing the problem dimensionality, then dividing the decision variables into the all-zero group, all-one group, and mixed group, which can significantly reduce the dimensionality of the decision space.

Based on the review of existing work, it can be observed that these methods extract the features of the population in the decision space [37, 51] or the features of problem datasets [25] as the states for reinforcement learning, which are ineffective for solving large-scale SMOPs due to the curse of dimensionality, i.e., there exist too many features to be learned [4, 24]. Besides, these methods are not tailored for selecting the best guiding vector for solving large-scale SMOPs.

2.4. Motivation of this work

According to the above analysis, it can be found that existing MOEAs cannot adaptively select the best guiding vector for solving large-scale SMOPs. Besides, the curse of dimensionality poses challenges in utilizing existing deep reinforcement learning assisted evolutionary algorithms for guiding vector selection [4, 24]. To address the above challenges, the motivation of this work is to develop new techniques for the effective selection of guiding vectors during the evolutionary process.

DQN (Deep Q-Network) [29] is an algorithm based on deep learning and reinforcement learning, used to address action selection problems in Markov Decision Processes (MDPs). It has been employed to tackle reinforcement learning tasks in discrete action spaces. As the selection of guiding vectors can also be considered as a discrete action space, in our work, we employ DQN for the selection of guiding vectors. In this work, we propose an evolutionary algorithm that incorporates DQN-based automated guiding vector selection method and a novel adaptive clustering method. To be specific, the proposed algorithm can automatically recommend a guiding vector according to the current population state by using the mapping model between the population state and the guiding vector. Moreover, based on the guiding vector recommended by the deep neural network, all the decision variables are clustered into two groups. The larger group can be used to significantly reduce the number of decision variables, while the smaller group can finely search for the optimal values of non-zero decision variables. Compared with existing sparse MOEAs that divide decision variables into a number of groups, the proposed algorithm can recommend the best guiding vector for the current distribution and more quickly approximate sparse Pareto optimal solutions, thus achieving a better generalization performance. In the next section, the details of the proposed algorithm are elaborated.

3. The proposed algorithm

In this section, we first present the main framework of the proposed AGSEA, then provide a detailed explanation of its three important components as illustrated in Fig. 2, including the automated guiding vector selection method, the adaptive variable clustering method, and the novel genetic operators.

3.1. Procedure of AGSEA

Algorithm 1 presents the pseudocode of the main framework of the proposed AGSEA. To begin with, N solutions are generated to form the initial population (Line 4). At each generation, the reinforcement learning agent selects a guiding vector (i.e., action) for the generation of offspring solutions (Line 8). Next, the binary tournament selection strategy is applied to select $2N$ parent solutions from the current P (Line 9) based on a fitness evaluated according to dominance relations and Euclidean distances. Then, all decision variables are divided into two groups by using the proposed adaptive variable clustering method, which assists the generation of offspring solutions Q, by means of the proposed genetic operators (Line 10). Afterwards, the environment selection method of SPEA2 (Line 13) is used to select N solutions from the combined population of P and Q. Moreover, the solutions and Hypervolume (HV) [49] improvement (i.e., reward) are recorded to the experience memory pool M for training the deep neural network Net. Finally, P is returned as the final population when the termination condition is met.

In order to effectively generate sparse solutions, we use the bi-level encoding scheme as suggested in [44]. To be specific, each solution $\mathbf{x}$ is represented by a real vector **real** and a binary vector **bin** in the proposed AGSEA, where $x_i = real_i \times bin_i$ for $i = 1, \ldots, d$. As shown in Algorithm 2, AGSEA initializes the population based on the above encoding scheme, where the initial solutions are randomly generated with different degrees of sparsity.

Algorithm 1: Main procedure of AGSEA

Input: N (population size), FE_{max} (maximum number of evaluations), S (number of samplings)
Output: P (final population)

1 $Net \leftarrow$ Randomly initialize a deep neural network;
2 $M \leftarrow \emptyset$; //Experience memory pool
3 $Action \leftarrow 1$; //Initial action
4 $[P, \mathbf{gv1}] \leftarrow Initialization(S)$; //Algorithm 2
5 $FE \leftarrow |P|$; //Number of consumed evaluations
6 **while** $FE \leq FE_{max}$ **do**
7 $\quad Q_{current} \leftarrow Q$;
8 $\quad \mathbf{gv} \leftarrow GVSelection(Net, M)$; //Algorithm 4
9 $\quad P' \leftarrow$ Select $2N$ parents from P via the mating selection strategy of SPEA2;
10 $\quad Q \leftarrow Variation(P', \mathbf{gv})$; //Algorithm 5
11 $\quad FE \leftarrow FE + |Q|$;
12 $\quad P \leftarrow$ Select N solutions from $P \cup Q$ via the environmental selection strategy of SPEA2;
13 $\quad M \leftarrow$ Insert the current population states, rewards, and selected actions to M;
14 $\quad Net \leftarrow Training(Net, M)$; //Algorithm 3
15 **return** P;

3.2. Automated guiding vector selection

Different guiding vectors can direct genetic operators to generate offspring solutions with corresponding sparse preferences when solving large-scale sparse SMOPs. In order to achieve a better balance between exploration and exploitation, this paper designs three different guiding vectors to efficiently generate sparse offspring solutions during the evolutionary process.

The first guiding vector **gv1** is the priori guiding vector, which seeks to analyze the a priori fitness contribution of each decision variable to the objective functions before the iteration process. The calculation process of **gv1** is shown in Algorithm 2 (Lines 2–12), where a larger value in **gv1** indicates that the corresponding decision variable contributes less to the objective functions. To be specific, a matrix $C \in \mathbb{R}^{S \times d}$ is firstly generated, whose i-th column is sampled S times by the Latin hypercube sampling method within the range of the i-th decision variable. When traversing each row of matrix C, a matrix $R \in \mathbb{R}^{d \times d}$ is generated, whose each row is the i-th row of matrix C. Then, a $d \times d$ identity matrix B is generated, and a matrix $X \in \mathbb{R}^{d \times d}$ is obtained with $X_{ji} = R_{ji} \times B_{ji}$. By regarding each row of X as a solution, a population Q is obtained, where all the decision variables of the i-th solution are set to zero except the i-th decision variable one. The L1-norm of the d solutions in Q are calculated and stored in **gv1**.

The second guiding vector **gv2** is calculated based on the information of the current population. The sparse distribution of the current population can represent the evolution direction towards optimal sparsity. The higher the frequency of non-zero decision variables in a particular dimension of the current population, the higher the probability of that dimension being non-zero in the future evolution process. Therefore, this paper extracts a statistical guiding vector (i.e., the second guiding vector **gv2**) based on a simple yet efficient voting mechanism from the current population using the following formula:

$$gv2_i = \frac{1}{|P|} \sum_{\mathbf{p} \in P} \mathbf{bin}_i^p , \tag{2}$$

where $gv2_i$ is the sparsity of the i-th variable and $\mathbf{bin}_i^p$ is the i-th variable of the binary vector of solution $\mathbf{p}$ in population P.

The guiding vectors **gv1** and **gv2** use the Latin hypercube sampling method and statistical methods to indicate the importance of each decision variable. While this can guide the population to quickly approximate the sparse Pareto front, it can only explore a part of the decision space. As a consequence, to increase the search ability of the proposed algorithm, we introduce the third guiding vector **gv3** whose

Algorithm 2: $Initialization(N, S)$

Input: N (population size), S (number of samplings)
Output: P (initial population), **gv1** (first guiding vector)

$d \leftarrow$ Number of decision variables;
$C \leftarrow$ Generate S values for each decision variable by the Latin hypercube sampling;
$P \leftarrow \emptyset$; `//Initial population`
gv1 $\leftarrow 1 \times d$ vector of zeros;
for $s = 1, \ldots, S$ **do**
  $R \leftarrow d \times d$ matrix generated;
  $B \leftarrow d \times d$ identity matrix;
  $Q \leftarrow$ Generate a population with decision vectors taken from $R \cdot B$;
  for $i = 1, \ldots, d$ **do**
    $L1_i \leftarrow$ The L1-norm of the i-th solution in Q;
    $gv1_i \leftarrow gv1_i + L1_i$;
  $P \leftarrow P \cup Q$;
`//Generate the initial population`
$R \leftarrow$ Uniformly randomly generate the decision variables of N solutions;
$B \leftarrow N \times D$ matrix of zeros;
for $i = 1$ *to* N **do**
  for $j = 1$ *to* $rand() \times D$ **do**
    $[m, n] \leftarrow$ Randomly select two decision variables;
    if $gv1_m < gv1_n$ **then**
      Set the m-th element in the i-th row of B to 1;
    else
      Set the n-th element in the i-th row of B to 1;
$Q \leftarrow$ A population whose i-th solution is generated by the $R \cdot B$;
$P \leftarrow N$ solutions are chose from $P \cup Q$ via the environment selection strategy of SPEA2;
return P and **gv1**;

each element is randomly sampled within $[0, 1]$ during the evolutionary process. **gv3** can randomly represent the importance of each decision variable, so each variable has an equal probability of being flipped to zero or non-zero, further enhancing the exploration capability of the decision space, thus reducing the probability of the proposed algorithm falling into local optima.

To adaptively select the most appropriate guiding vector for the current population state, we establish a mapping model between the population state and the above three guiding vectors (i.e., **gv1**, **gv2**, and **gv3**) using reinforcement learning. The core problem in using reinforcement learning is determining the rewards and states associated with the selected guiding vector, so that the agent can learn the relationship between population states and the expected

Algorithm 3: $Training(Net, M)$

Input: Q (the deep neural network), M (experience memory pool), λ (ratio of consumed evaluations)

if $\lambda < 0.25$ **then**
  No training;
else if $\lambda = 0.25$ **then**
  Use all samples in M to train DQN;
else
  Use the last ten samples to update the DQN every ten iterations;
Output: Net (the deep neural network)
return Net;

Algorithm 4: $GV Selection(Net, M)$

Input: M (experience memory pool), λ (ratio of consumed evaluations)
Output: **gv** (recommended guiding vector)

$r \leftarrow$ Generate a random value between 0 and 1;
if $\lambda < 0.25$ *or* $r < 0.3^{\lambda}$ **then**
  gv $\leftarrow$ Randomly select a guiding vector;
else
  Select a guiding vector using DQN;
return gv;

cumulative rewards of guiding vectors using a deep neural network. The states and rewards are defined as follows:

States. In order to accurately reflect the sparse distribution of the current population, we extract the following features from the current population as the states: average sparsity, standard deviation of solutions' sparsity, the proportion of solutions in each sparsity interval (a total of 10 equal-sized intervals are divided sequentially within $[0, 1]$), and the ratio of consumed function evaluations. Compared with directly input decision variables as the states, the designed states in this paper can better reflect the sparse distribution of the current population. Through these features, the current diversity and sparsity characteristics of the population can be comprehensively reflected to guide decision-making by the reinforcement learning agent. For example, when the diversity of the population is poor, the sparse distribution of the population will be more concentrated, and the standard deviation of the sparse distribution will be smaller. In this case, choosing the random guiding vector can improve the diversity of the population.

Rewards. The hypervolume is used to comprehensively assess the performance of MOEAs, which evaluates the quality of a solution set by measuring the hypervolume occupied by the set. As a consequence, the improvement in HV value of the current population relative to the previous generation is used as the reward for selecting the guiding vector, which not only reflects the convergence but also the diversity improvement of the population.

Algorithm 3 shows the training process of DQN. When a quarter of the total function evaluation has been consumed, the neural network is trained using gradient descent on the loss

$$\mathcal{L} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left(Net\left(s_t, a_t\right) - q_t \right)^2 , \tag{3}$$

where $\mathcal{T}$ is all the samples in the experience pool M, $Net\left(s_t, a_t\right)$ is the predicted value of the action a_t for the state s_t, and r_t is the corresponding reward at the i-th generation. Later in the evolutionary process, the latest ten samples from the experience pool will be used to update the neural network every ten iterations based on Eq. (3), and

$$q_t = r_t + \gamma \max_{a' \in \mathcal{A}} Net(s_{t+1}, a') . \tag{4}$$

It is worth to note the expected output q_t contains not only the current reward r_t but also the maximum reward $\max_{a' \in \mathcal{A}} Net(s_{t+1}, a')$ obtained by taking the next action a' and the $\gamma \in [0, 1]$ is the discount factor. In addition, Epsilon's greedy strategy is used to mitigate the negative impact of DQN overfitting as shown in Algorithm 4, which randomly selects a guiding vector from all the guiding vectors with a probability of ϵ to replace DQN:

$$a_t = \begin{cases} rand\_action & if\ rand < \epsilon \\ \underset{a' \in \mathcal{A}}{\operatorname{argmin}} \max Net(s_{t+1}, a') & otherwise \end{cases} , \tag{5}$$

where *rand_action* is a random action and *rand* is a random number belonging to $[0, 1]$. If the guiding vector is used for the first time, the initialized action is used. In addition, considering that the proposed AGSEA needs to obtain more random samples early on, we set ϵ to 0.3^{λ} (λ is the ratio of consumed evaluations), which can enhance the exploration ability in the early evolutionary process.

3.3. Adaptive variable clustering

The proposed adaptive variable clustering method divides all decision variables into two groups using K-means clustering according to the selected guiding vector, which guides the generation of offspring solutions. Since the selected guiding vector directly reflects the critical degree of each decision variable, the critical variables and non-critical variables can be grouped into different groups. By flipping decision variables at different granularity, the larger group can quickly reduce the dimensionality of the decision space, and the smaller group can fine-adjust the value of more critical variables.

It is worth noting that the dimensionality reduction using variable clustering for solving large-scale SMOPs has been suggested in existing sparse MOEAs [50], however, the idea of the proposed adaptive variable clustering method is essentially different. Firstly, current MOEAs partition decision variables into numerous groups, posing challenges in dealing with a large number of groups. In contrast, the proposed grouping method statically divides decision variables into two groups, leading to a more significant dimensionality reduction effect. Secondly, the clustering method presented in this paper can adapt to different guiding vectors, thereby achieving a better balance between exploitation and exploration.

Algorithm 5: $Variation(P', \mathbf{gv})$

Input: P' (parent solutions), $\mathbf{gv}$ (guiding vector)
Output: Q (offspring population)

$Groups \leftarrow$ Use K-means clustering to divide all decision variables into two groups based on $\mathbf{gv}$;
while $P' \neq \emptyset$ **do**
  $[\mathbf{p}, \mathbf{q}] \leftarrow$ Randomly select two parents from P';
  $group \leftarrow$ Randomly select a group from $Groups$;
  $index \leftarrow xor(\mathbf{bin}^p, \mathbf{bin}^q)$;
  $P' \leftarrow P' \setminus \{\mathbf{p}, \mathbf{q}\}$;
  //Generate the *mask* of offspring $\mathbf{o}$
  $\mathbf{bin}^o \leftarrow \mathbf{bin}^p$; //$\mathbf{bin}^p$ denotes the binary vector *mask* of solution $\mathbf{p}$
  //Crossover
  $index \leftarrow index \cap group$;
  if $rand() < 0.5$ **then**
    $\mathbf{bin}^o_{index} \leftarrow 1$;
  else
    $\mathbf{bin}^o_{index} \leftarrow 0$;
  $prob \leftarrow$ Calculate the mutation probability by Eq. (6) //Mutation
  if $rand() < 0.5$ **then**
    $rand_d \leftarrow$ Generate d random numbers in $[0, 1]$;
    $index \leftarrow$ Elements in $rand_d$ that are less than $prob$;
    $\mathbf{bin}^o_{index} \leftarrow 1$;
  else
    $rand_d \leftarrow$ Generate d random numbers in $[0, 1]$;
    $index \leftarrow$ Elements in $rand_d$ that are less than $prob$;
    $\mathbf{bin}^o_{index} \leftarrow 0$;
  //Crossover and mutation for real vectors
  $\mathbf{real}^o \leftarrow$ Perform simulated binary crossover and polynomial mutation on $\mathbf{bin}^{\mathbf{p}}$ and $\mathbf{bin}^q$;
  $\mathbf{o} \leftarrow$ A solution determined by $\mathbf{bin}^o$ and $\mathbf{real}^o$;
  $Q \leftarrow Q \cup \{\mathbf{o}\}$;
return Q;

3.4. Genetic operators

In order to use the clustering outcomes to enhance search ability, the clustering outcomes is embedded in the genetic operators. As described in Algorithm 5, two parent solutions $\mathbf{p}$ and $\mathbf{q}$ are selected from P' to generate an offspring solution $\mathbf{o}$ and *group* is randomly selected from two variable groups. Then, the crossover operator sets the *mask* of offspring $\mathbf{o}$ to

the same as the parent $\mathbf{p}$, and the following two operations are performed with equal probability: (1) Setting these decision variables in $index \cap group$ to one; (2) setting these decision variables in $index \cap group$ to zero. By doing so, many variables can be simultaneously flipped.

In addition, in order to enhance the diversity of offspring solutions efficiently, an adaptive mutation probability is calculated by

$$prob = \frac{\sum_{i=1}^{d} \sum_{\mathbf{p} \in P} \mathbf{bin}_i^p}{(d * GVNum)^2}, \tag{6}$$

where $\sum_{i=1}^{d} \sum_{\mathbf{p} \in P} \mathbf{bin}_i^p$ is the average value of $\mathbf{bin}$ in the current population, $GVNum$ is the number of different guide vectors used in the last ten iterations, and d is the number of decision variables. On the one hand, if most decision variables in the current population are non-zero variables, the mutation probability will increase so that more decision variables can be flipped to zero. On the other hand, if the proposed reinforcement learning agent always recommends the same guiding vector due to overfitting, the mutation probability will increase to enhance the diversity of offspring solutions. Then a $1 \times d$ vector $rand_d$is generated, whose each element is randomly sampled from 0 to 1, and the index of elements smaller than $prob$ is recorded in the index. The mutation process of $mask$ of $\mathbf{o}$ involves the same probability of undergoing one of the following two operations: (1) Setting these decision variables in $index$ to one; (2) Setting these decision variables in $index$ to zero. Lastly, the real vector dec of $\mathbf{o}$ is generated as suggested in [48].

4. Empirical studies

In this section, we verify the performance of the proposed AGSEA on eight benchmark problems SMOP1–SMOP8 and three real-world SMOPs, namely, pattern mining [1], feature selection [3] and critical node detection [31]. DGEA [16], SparseEA2 [48], TS-SparseEA [19], DSGEA [50], and S-NSGA-II [22] are selected as baselines in the experiments, where DGEA is a representative large-scale MOEA based on adaptive offspring generation strategy, SparseEA2 and S-NSGA-II are state-of-the-art sparse MOEAs based on new variation operators, TS-SparseEA is a state-of-the-art sparse MOEA based on dimensionality reduction, and DSGEA is a state-of-the-art sparse MOEA based on variable clustering.

4.1. Settings of algorithms

For fair comparisons, the population size of all the algorithms is set to 100 on all problems. All the algorithms are run for a maximum of $100 \times d$ function evaluations on benchmark problems and 100 000 function evaluations on real-world problems, respectively. All compared algorithms adopt the recommended parameter values in their original literature. For DGEA, the number of environmental selection operation is set to 1, and the number of reference vectorss for offspring generation is set to 10. For SparseEA2, the number of groups is set to 4. For TS-SparseEA, the ratio of function evaluations is set to 0.1 for the first stage, and the number of groups is set to 50. For the proposed AGSEA, the number of samplings is set to 5 and the discount factor is set to 0.1. Besides, DGEA and S-NSGA-II generate real variables within [0, 1] and round them to obtain binary decision variables when solving binary SMOPs. For the neural network that are used in this paper, it is structured as a fully connected neural network with three hidden layers, each consisting of 10 nodes, and employs the Sigmoid function as the activation function for each hidden layer.

4.2. Settings of problems

Table 1 provides the detailed parameter settings for both benchmark problems and real-world applications. In this table, the number of objectives m is set to 2 and the number of decision variables d varies from 100 to 5 000. Additionally, the sparsity of Pareto optimal solutions is set to 0.1, which indicates the proportion of non-zero decision variables among all decision variables. For real-world applications, their sparsity is unknown and cannot be specified. The benchmark problems SMOP1–SMOP8 have different challenges like low intrinsic dimensionality, deception, and multimodality, and the real-world applications have complex landscapes with binary decision spaces. Inverted generational distance (IGD) [2] and hypervolume (HV) [49] are employed to assess the population quality on benchmark problems and real-world applications, respectively. Besides, Wilcoxon's test [10] at the 0.05 level is employed to estimate the significance of the difference between each compared algorithm and the proposed AGSEA according to 30 independent runs on each optimization problem. "+" or "−" indicates that the compared algorithm is significantly better or worse than AGSEA, respectively. "=" shows that the proposed AGSEA and the compared algorithm have a similar performance.

4.3. Results on benchmark problems

Table 2 presents the statistical results of the IGD values of the six MOEAs on the benchmark problems SMOP1–SMOP8 with 100 to 5 000 decision variables obtained via 30 independent runs, where the best result on each benchmark problem is shown in a gray background. From the table, it is obvious the general large-scale MOEA (i.e., DGEA) is unsuitable for solving SMOPs because it does not take into account the sparse nature of the SMOPs. As a contrast, the sparse MOEAs (i.e., SparseEA2, TS-SparseEA, DSGEA, S-NSGA-II) can achieve a promising performance on SMOPs, but their performance is worse than the proposed AGSEA. In terms of statistical tests on 40 test instances, AGSEA significantly outperforms SparseEA2, TS-SparseEA, DSGEA, S-NSGA-II on 33, 28, 29, and 28 test instances, respectively.

Fig. 3 shows the parallel coordinates plot [26] of the decision variables of final solutions obtained by the compared MOEAs and the proposed AGSEA. For DGEA, it cannot obtain sparse solutions on SMOPs according to Fig. 3. As for sparse MOEAs including the proposed AGSEA, most variables of the obtained solutions are zero, indicating that

Table 1
Parameter settings of eight benchmark SMOPs and three practical SMOPs.

Benchmark problem	Type of variables	No. of variables d	No. of objective m	Sparsity of Pareto optimal solutions (ratio of nonzero variables in optimal solutions)
SMOP1–SMOP8	Real	100 500 1 000 5 000	2	0.1

Critical node detection	Type of variables	No. of variables d	Dataset	No. of nodes	No. of edges
CN1	Binary	102	Hollywood Film Music[1]	311	640
CN2		452	GD (A99)[1]	234	154
CN3		466	GD (A01)[1]	311	640
CN4		500	GD (C97)[1]	452	460

Feature selection	Type of variables	No. of variables	Dataset	No. of samples	No. of features	No. of classes
FS1	Binary	166	MUSK1[2]	476	166	2
FS2		256	Semeion Handwritten Digit[2]	1593	256	10
FS3		310	LSVT Voice rehabilitation[2]	126	310	2

Pattern mining problem	Type of variables	No. of variables	Dataset	No. of transactions	No. of items	Avg. length of transactions
PM1	Binary	100	Synthetic [1]	10000	100	50
PM2		500	Synthetic [1]	10000	500	50
PM3		1000	Synthetic [1]	10000	1000	50
PM4		2000	Synthetic [1]	10000	2000	50

[1] http://vlado.fmf.uni-lj.si/pub/networks/data/default.html
[2] http://archive.ics.uci.edu/ml/index.php

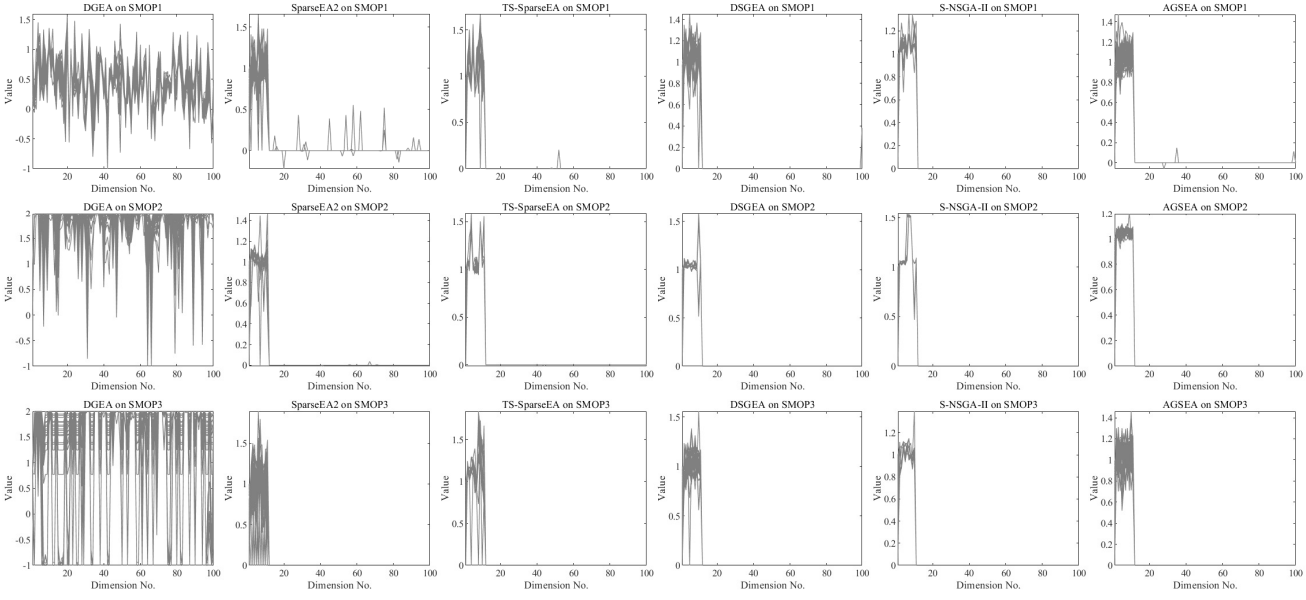

Figure 3: Parallel coordinates of the decision variables obtained by DGEA, SparseEA2, TS-SparseEA, S-NSGA-II and the proposed AGSEA on SMOP1, SMOP2, and SMOP3 with 100 decision variables.

existing sparse MOEAs are more suitable for solving large-scale SMOPs. To further differentiate the performance between the four compared sparse MOEAs and the proposed AGSEA, Fig. 4 illustrates the convergence profiles of IGD values obtained by them on SMOP1, SMOP2, and SMOP3 with 500 decision variables. The results demonstrate that AGSEA converges faster than SparseEA2, TS-SparseEA, DSGEA, and S-NSGA-II. Notably, even with only 5 000 function evaluations, the populations generated by AGSEA are competitive with those produced by SparseEA2, TS-SparseEA, DSGEA, and S-NSGA-II with 10 000 function evaluations on SMOP1, SMOP2, and SMOP3. In addition, the performance of the proposed AGSEA falls short on SMOP5–8, primarily due to the more complex function landscapes of these problems (sparse position uncertainty in SMOP5 and SMOP6, and function landscapes with epistasis

Table 2
IGD values obtained by DGEA, SparseEA2, TS-SparseEA, S-NSGA-II and the proposed AGSEA on SMOP1–SMOP8 with 100 to 5 000 decision variables.

Problem	d	DGEA	SparseEA2	TS-SparseEA	DSGEA	S-NSGA-II	AGSEA
SMOP1	100	6.5232e-1 (4.67e-2) −	5.6771e-3 (5.31e-4) −	6.2976e-3 (2.35e-3) −	4.6679e-3 (2.87e-4) −	5.2772e-3 (1.51e-3) −	3.8555e-3 (7.07e-5)
	500	7.2477e-1 (3.29e-2) −	6.8001e-3 (6.16e-4) −	6.5914e-3 (1.51e-3) −	5.8029e-3 (8.86e-4) −	7.9633e-3 (3.61e-3) −	4.0848e-3 (1.14e-4)
	1000	7.2838e-1 (3.42e-2) −	7.2669e-3 (5.49e-4) −	8.2034e-3 (1.39e-3) −	6.2591e-3 (1.09e-3) −	1.0942e-2 (3.71e-3) −	4.2661e-3 (1.23e-4)
	2000	7.7102e-1 (2.82e-2) −	1.2545e-2 (1.41e-3) −	1.0546e-2 (1.98e-3) −	8.2570e-3 (1.40e-3) −	1.5039e-2 (5.32e-3) −	4.3163e-3 (1.23e-4)
	5000	7.7847e-1 (4.25e-2) −	3.2112e-2 (1.46e-3) −	1.3238e-2 (2.22e-3) −	3.3882e-2 (5.35e-3) −	1.9996e-2 (3.70e-3) −	4.3481e-3 (1.40e-4)
SMOP2	100	1.5526e+0 (1.77e-1) −	8.9383e-3 (2.58e-3) −	1.5987e-2 (6.30e-3) −	7.1904e-3 (2.88e-3) −	1.4815e-2 (7.43e-3) −	4.3533e-3 (3.21e-4)
	500	1.4129e+0 (8.75e-3) −	1.1516e-2 (3.61e-3) −	2.0459e-2 (5.12e-3) −	1.7885e-2 (3.99e-3) −	3.2660e-2 (1.26e-2) −	4.1144e-3 (3.80e-4)
	1000	1.4091e+0 (6.00e-3) −	1.3774e-2 (3.71e-3) −	2.7139e-2 (7.28e-3) −	1.8420e-2 (2.98e-3) −	4.4386e-2 (1.29e-2) −	4.5973e-3 (2.36e-4)
	2000	1.4092e+0 (3.78e-3) −	2.8718e-2 (3.24e-3) −	3.6991e-2 (5.11e-3) −	1.9363e-2 (2.73e-3) −	6.3286e-2 (1.27e-2) −	4.8591e-3 (5.59e-4)
	5000	1.4063e+0 (4.31e-3) −	7.3960e-2 (3.40e-3) −	5.7777e-2 (6.90e-3) −	3.6968e-2 (6.96e-3) −	8.7242e-2 (1.52e-2) −	4.9648e-3 (3.92e-4)
SMOP3	100	1.3788e+0 (2.49e-1) −	7.1156e-3 (1.74e-3) −	1.4047e-2 (2.30e-2) −	4.8186e-3 (8.54e-4) −	9.0721e-3 (4.10e-3) −	3.8437e-3 (5.90e-5)
	500	1.0999e+0 (3.99e-1) −	8.2708e-3 (1.72e-3) −	7.9743e-3 (2.75e-3) −	5.2202e-3 (5.84e-4) −	1.1291e-2 (4.15e-3) −	3.9080e-3 (7.65e-5)
	1000	8.3133e-1 (2.95e-1) −	1.1242e-2 (1.54e-3) −	6.8591e-3 (1.79e-3) −	5.6987e-3 (7.35e-4) −	1.7384e-2 (5.22e-3) −	3.9739e-3 (1.09e-4)
	2000	8.8530e-1 (3.40e-1) −	1.8364e-2 (2.00e-3) −	6.1896e-3 (1.65e-3) −	6.7921e-3 (1.35e-3) −	2.1432e-2 (3.92e-3) −	3.9777e-3 (7.76e-5)
	5000	9.1034e-1 (3.54e-1) −	3.2204e-2 (1.51e-3) −	6.6680e-3 (1.28e-3) −	2.2423e-2 (3.51e-3) −	2.3017e-2 (4.07e-3) −	4.0332e-3 (1.36e-4)
SMOP4	100	7.8595e-1 (9.78e-2) −	4.7191e-3 (2.62e-4) −	4.6274e-3 (1.86e-4) −	4.7650e-3 (3.59e-4) −	5.3462e-3 (4.10e-4) −	4.1540e-3 (6.40e-5)
	500	7.0094e-1 (1.52e-2) −	4.7842e-3 (2.49e-4) −	4.5570e-3 (1.26e-4) −	4.8116e-3 (2.37e-4) −	4.8341e-3 (3.21e-4) −	4.1406e-3 (7.08e-5)
	1000	6.9979e-1 (1.96e-2) −	4.7507e-3 (2.43e-4) −	4.6204e-3 (1.31e-4) −	4.7202e-3 (2.52e-4) −	4.8725e-3 (3.00e-4) −	4.1311e-3 (6.70e-5)
	2000	7.0036e-1 (1.34e-2) −	4.7580e-3 (2.50e-4) −	4.6327e-3 (2.15e-4) −	4.7575e-3 (2.21e-4) −	4.8425e-3 (2.46e-4) −	4.1222e-3 (7.30e-5)
	5000	7.0028e-1 (1.53e-2) −	4.7745e-3 (2.58e-4) −	4.6501e-3 (1.71e-4) −	4.7985e-3 (2.51e-4) −	4.8713e-3 (3.90e-4) −	4.1518e-3 (6.96e-5)
SMOP5	100	4.4059e-1 (1.08e-2) −	5.6712e-3 (2.97e-4) +	6.3357e-3 (6.34e-4) ≈	6.7479e-3 (4.55e-4) −	4.7956e-3 (2.13e-4) +	6.3048e-3 (7.20e-4)
	500	4.5288e-1 (1.64e-2) −	5.2830e-3 (1.62e-4) −	5.2514e-3 (2.80e-4) −	5.0369e-3 (2.09e-4) −	4.7844e-3 (2.40e-4) +	4.9032e-3 (1.53e-4)
	1000	4.5553e-1 (1.83e-2) −	5.2676e-3 (1.78e-4) −	5.1397e-3 (1.71e-4) −	5.0428e-3 (1.95e-4) −	4.9349e-3 (4.12e-4) ≈	4.9050e-3 (1.80e-4)
	2000	4.4322e-1 (2.61e-2) −	5.1672e-3 (2.19e-4) −	5.2447e-3 (2.26e-4) −	5.1447e-3 (2.68e-4) −	8.6908e-3 (3.44e-3) −	4.7328e-3 (1.20e-4)
	5000	4.2676e-1 (2.49e-2) −	5.1005e-3 (1.97e-4) −	5.4430e-3 (2.02e-4) −	5.6169e-3 (4.76e-4) −	2.0749e-2 (2.47e-3) −	4.5429e-3 (1.03e-4)
SMOP6	100	1.8868e-1 (3.42e-2) −	6.7955e-3 (4.02e-4) +	6.7031e-3 (8.13e-4) +	6.6129e-3 (7.24e-4) +	4.8371e-3 (2.02e-4) +	7.3522e-3 (1.01e-3)
	500	1.9452e-1 (5.48e-2) −	6.4727e-3 (2.43e-4) −	5.3617e-3 (2.33e-4) +	5.2815e-3 (2.66e-4) +	4.7959e-3 (2.62e-4) +	5.6073e-3 (2.84e-4)
	1000	1.8975e-1 (6.26e-2) −	6.6835e-3 (3.37e-4) −	5.3190e-3 (2.24e-4) +	5.3164e-3 (1.95e-4) +	4.9102e-3 (1.96e-4) +	5.7711e-3 (2.75e-4)
	2000	1.7413e-1 (6.44e-2) −	6.7866e-3 (3.71e-4) −	5.4202e-3 (1.73e-4) +	5.6316e-3 (2.49e-4) +	4.8729e-3 (1.95e-4) +	5.8594e-3 (2.35e-4)
	5000	1.4736e-1 (5.61e-2) −	6.9550e-3 (4.31e-4) −	5.8156e-3 (1.87e-4) +	6.4738e-3 (1.95e-4) −	8.3988e-3 (3.45e-3) −	6.0489e-3 (3.19e-4)
SMOP7	100	7.2073e-1 (1.24e-1) −	1.3257e-2 (4.70e-3) −	2.5659e-2 (1.27e-2) −	9.4154e-3 (8.13e-3) −	1.8952e-2 (1.02e-2) −	6.6944e-3 (4.73e-3)
	500	5.5027e-1 (5.79e-2) −	8.4889e-3 (2.29e-3) −	3.2583e-2 (7.40e-3) −	2.5251e-2 (6.91e-3) −	4.6065e-2 (1.96e-2) −	6.3630e-3 (2.12e-3)
	1000	5.3357e-1 (1.46e-1) −	9.4016e-3 (1.61e-3) +	3.9781e-2 (6.71e-3) −	2.7423e-2 (3.60e-3) −	6.8327e-2 (2.52e-2) −	1.1862e-2 (3.43e-3)
	2000	5.7956e-1 (1.11e-1) −	1.5161e-2 (1.94e-3) +	5.4217e-2 (7.25e-3) −	2.8837e-2 (4.21e-3) ≈	1.0137e-1 (2.17e-2) −	3.0965e-2 (5.04e-3)
	5000	5.3544e-1 (1.50e-1) −	6.2926e-2 (3.35e-3) +	7.4103e-2 (8.40e-3) +	3.2009e-2 (2.51e-3) +	1.3561e-1 (2.31e-2) −	8.7541e-2 (1.41e-2)
SMOP8	100	3.1215e+0 (9.22e-2) −	1.4437e-1 (2.26e-2) −	1.2996e-1 (3.64e-2) ≈	1.0611e-1 (2.32e-2) +	1.1292e-1 (3.60e-2) +	1.2515e-1 (2.49e-2)
	500	2.4575e+0 (8.74e-1) −	1.7829e-1 (1.11e-2) −	1.4181e-1 (2.42e-2) +	1.5626e-1 (1.46e-2) ≈	1.5049e-1 (1.76e-2) +	1.6150e-1 (1.20e-2)
	1000	2.2309e+0 (8.86e-1) −	2.0193e-1 (1.22e-2) −	1.4199e-1 (1.66e-2) +	1.6224e-1 (1.37e-2) +	1.7041e-1 (1.89e-2) +	1.8483e-1 (1.11e-2)
	2000	2.6314e+0 (5.76e-1) −	2.1604e-1 (7.10e-3) +	1.4139e-1 (9.04e-3) +	1.7554e-1 (9.28e-3) +	1.9586e-1 (1.83e-2) +	2.2924e-1 (1.14e-2)
	5000	2.5207e+0 (7.59e-1) −	2.5060e-1 (5.81e-3) +	1.6580e-1 (6.83e-3) +	1.7703e-1 (6.68e-3) +	2.1414e-1 (1.96e-2) +	3.0363e-1 (1.49e-2)
+/ − / ≈		0/40/0	7/33/0	10/28/2	9/29/2	11/28/1	

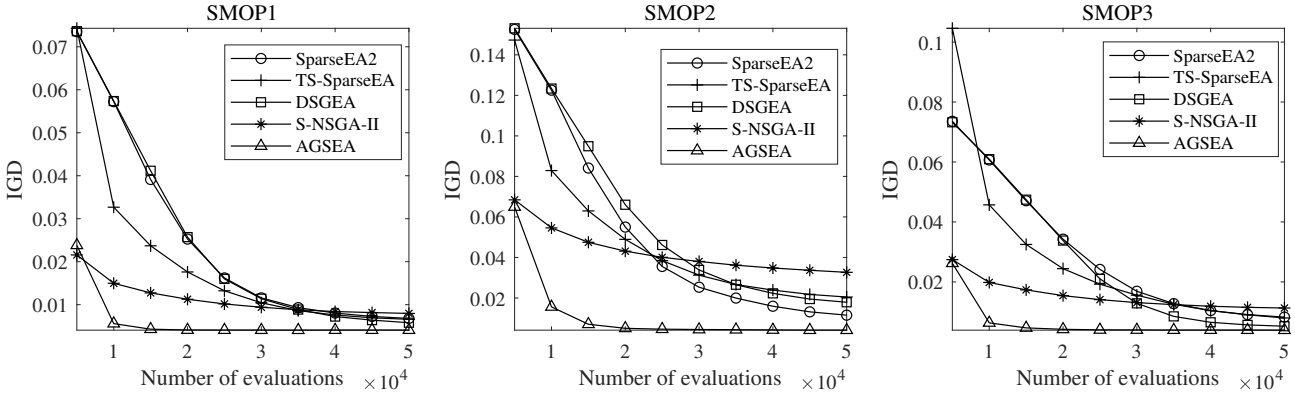

Figure 4: Convergence profiles of IGD values obtained by DGEA, SparseEA2, TS-SparseEA, S-NSGA-II and the proposed AGSEA on SMOP1, SMOP2, and SMOP3 with 500 decision variables.

Table 3
HV values obtained by DGEA, SparseEA2, TS-SparseEA, S-NSGA-II and the proposed AGSEA on the critical node detection problem (CN1–CN4), the feature selection problem (FS1–FS3), and the pattern mining problem (PM1–PM4).

Problem	d	DGEA	SparseEA2	TS-SparseEA	DSGEA	S-NSGA-II	AGSEA
CN1	102	8.7789e-1 (2.76e-2) −	9.3431e-1 (1.28e-3) −	9.3570e-1 (2.49e-4) +	9.3355e-1 (1.46e-3) −	8.8751e-1 (3.95e-2) −	9.3525e-1 (5.34e-4)
CN2	234	9.0104e-1 (4.96e-2) −	9.7500e-1 (6.13e-4) −	9.7578e-1 (1.30e-4) ≈	9.7432e-1 (6.16e-4) −	9.3508e-1 (1.12e-2) −	9.7582e-1 (1.62e-4)
CN3	311	8.3896e-1 (2.42e-2) −	9.3650e-1 (1.53e-3) −	9.2856e-1 (1.91e-3) −	9.3376e-1 (3.48e-3) −	7.3098e-1 (6.67e-2) −	9.3783e-1 (1.72e-3)
CN4	452	9.4424e-1 (3.42e-2) −	9.9616e-1 (5.95e-5) −	9.9622e-1 (1.07e-6) −	9.9614e-1 (8.70e-5) −	9.8943e-1 (4.72e-3) −	9.9622e-1 (4.33e-7)
FS1	166	8.5348e-1 (8.02e-2) −	9.7617e-1 (9.18e-3) −	9.8834e-1 (5.40e-3) −	9.7329e-1 (1.18e-2) −	9.3041e-1 (1.60e-2) −	9.9110e-1 (3.56e-3)
FS2	256	8.2666e-1 (6.39e-2) −	9.3286e-1 (6.83e-3) −	9.3396e-1 (4.35e-3) −	9.2641e-1 (6.85e-3) −	8.4933e-1 (2.06e-2) −	9.4124e-1 (4.98e-3)
FS3	310	8.1652e-1 (8.08e-2) −	9.8383e-1 (1.78e-2) −	9.9844e-1 (9.94e-5) ≈	9.9218e-1 (1.36e-2) −	9.7540e-1 (2.22e-2) −	9.9844e-1 (1.10e-4)
PM1	100	1.4355e-1 (4.82e-2) −	2.8957e-1 (7.27e-3) −	3.0611e-1 (2.24e-3) −	2.8841e-1 (6.81e-3) −	2.6246e-1 (7.73e-3) −	3.1101e-1 (1.95e-5)
PM2	500	8.2645e-3 (3.53e-18) −	1.6985e-1 (3.58e-3) −	2.0424e-1 (4.20e-3) −	1.6994e-1 (3.13e-3) −	1.2149e-1 (6.13e-3) −	2.0919e-1 (4.25e-3)
PM3	1000	8.2645e-3 (3.53e-18) −	1.6293e-1 (3.57e-3) −	1.9391e-1 (6.45e-3) ≈	1.6374e-1 (3.56e-3) −	1.0803e-1 (5.07e-3) −	1.9422e-1 (4.51e-3)
PM4	2000	8.2645e-3 (3.53e-18) −	1.5431e-1 (4.12e-3) −	1.7684e-1 (4.52e-3) +	1.5449e-1 (3.44e-3) −	1.0205e-1 (3.78e-3) −	1.7230e-1 (4.15e-3)
+/−/≈		0/11/0	0/11/0	2/6/3	0/11/0	0/11/0	

in SMOP7 and SMOP8), which are challenging for the proposed adaptive clustering. Therefore, further research into clustering methods is warranted.

4.4. Results on real-world applications

In this section, the proposed AGSEA is compared with the other five MOEAs on real-world applications. Table 3 lists the mean values and standard deviation of the HV values obtained by DGEA, SparseEA2, TS-SparseEA, DSGEA, S-NSGA-II, and the proposed AGSEA on the critical node detection problem (CN1–CN4), the feature selection problem (FS1–FS3), and the pattern mining problem (PM1–PM4), averaged over 30 runs. According to Table 3, it can be found that the performance of the proposed AGSEA is significantly better than the other compared MOEAs. To be specific, AGSEA obtains the best HV values on nine test instances, TS-SparseEA performs the best on three test instances, while DGEA, SparseEA2, DSGEA, and S-NSGA-II cannot gain any best result. Besides, Fig. 5 depicts the convergence profiles of HV values obtained on PM2, where the general large-scale MOEA (i.e., DGEA) converges much more slowly than the five sparse MOEAs (i.e., SparseEA2, TS-SparseEA, DSGEA, S-NSGA-II, and AGSEA), and AGSEA has a faster convergence speed than SparseEA2, TS-SparseEA, DSGEA, and S-NSGA-II. It is worth noting that AGSEA shows superior performance on real-world applications compared to benchmark problems. This is primarily attributed to the greater difficulty in determining sparse locations in real-world applications, whereas AGSEA, by simultaneously introducing multiple guiding vectors to determine these sparse locations, achieves better results.

Based on the above empirical results, we can conclude that the proposed AGSEA is more effective for solving large-scale SMOPs than state-of-the-art large-scale or sparse MOEAs.

4.5. Ablation studies

To further verify the effectiveness of the proposed guiding vector selection method, AGSEA is compared with its variants with a single guiding vector, so that the influence

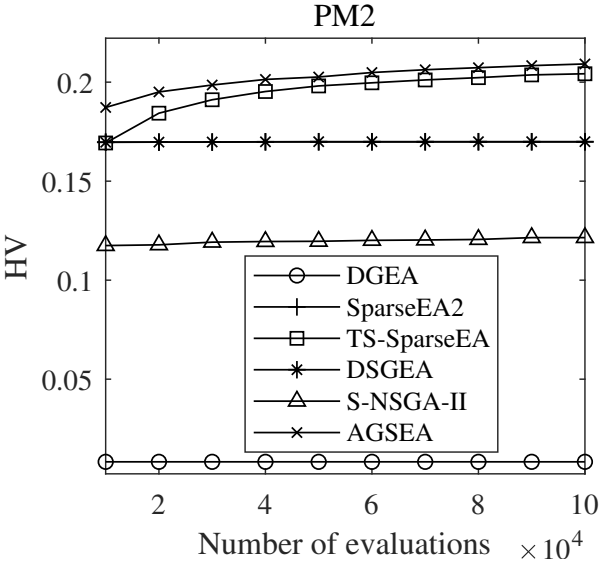

Figure 5: Convergence profiles of HV values obtained by DGEA, SparseEA2, TS-SparseEA, S-NSGA-II and the proposed AGSEA on PM2.

resulting from differences in other strategies can be completely eliminated. Table 4 lists the comparative results of AGSEA and its three variants, where AGSEA′, AGSEA″, and AGSEA‴ use only the guiding vector **gv1**, **gv2**, and **gv3**, respectively. It is obvious that the proposed AGSEA still exhibits the best overall performance, and is competitive to different variants on different SMOPs. Moreover, Fig. 6 presents the IGD values for the AGSEA and its variant AGSEA (rand) utilizing randomly selected guiding vectors, across the SMOP1–4. It is evident from the results that AGSEA continues to demonstrate superiority in performance. This is mainly because random selection can provide diverse guiding vectors, but does not take into account the characteristics of the current population. Fig. 7 depicts the ratio of guiding vectors selected by AGSEA during the optimization process of solving SMOP1, SMOP2, SMOP3, and SMOP4. It can be found that AGSEA does not

Table 4
IGD/HV values obtained by AGSEA′, AGSEA″, AGSEA‴ and AGSEA on SMOP1–SMOP8 with 1000 decision variables and real-world applications.

Problem	d	AGSEA′	AGSEA″	AGSEA‴	AGSEA
SMOP1	1000	4.2924e-3 (8.73e-5) −	3.5784e-2 (3.54e-3) −	2.0948e-2 (1.88e-3) −	4.2222e-3 (9.13e-5)
SMOP2	1000	4.7113e-3 (1.61e-4) −	7.2922e-2 (6.22e-3) −	4.3437e-2 (6.60e-3) −	4.6130e-3 (2.92e-4)
SMOP3	1000	4.2664e-3 (1.48e-4) −	3.1077e-2 (7.92e-3) −	3.0350e-2 (1.79e-3) −	3.9618e-3 (7.01e-5)
SMOP4	1000	4.1178e-3 (7.31e-5) ≈	4.1534e-3 (6.25e-5) ≈	4.1160e-3 (6.14e-5) +	4.1521e-3 (6.61e-5)
SMOP5	1000	4.8887e-3 (1.09e-4) ≈	5.0461e-3 (1.64e-4) −	5.8980e-3 (3.47e-4) −	4.8535e-3 (1.33e-4)
SMOP6	1000	6.6598e-3 (3.36e-4) −	6.2823e-3 (4.05e-4) −	7.4328e-3 (3.90e-4) −	5.7003e-3 (2.50e-4)
SMOP7	1000	7.6625e-2 (5.70e-3) −	8.3306e-2 (6.69e-3) −	7.3996e-2 (6.43e-3) −	1.1745e-2 (3.20e-3)
SMOP8	1000	2.2897e-1 (1.13e-2) −	2.3520e-1 (1.16e-2) −	2.1973e-1 (9.76e-3) −	1.8973e-1 (1.01e-2)
CN1	102	9.3252e-1 (9.19e-4) −	9.3205e-1 (1.61e-3) −	9.3046e-1 (1.16e-3) −	9.3519e-1 (5.61e-4)
FS1	166	9.9194e-1 (5.02e-4) +	9.9012e-1 (3.68e-3) −	9.9127e-1 (2.71e-3) ≈	9.9014e-1 (4.64e-3)
PM1	100	2.9792e-1 (3.33e-3) −	2.9606e-1 (4.68e-3) −	2.9602e-1 (4.00e-3) −	3.1097e-1 (2.37e-4)
+/ − / ≈		1/8/2	0/10/1	1/9/1	

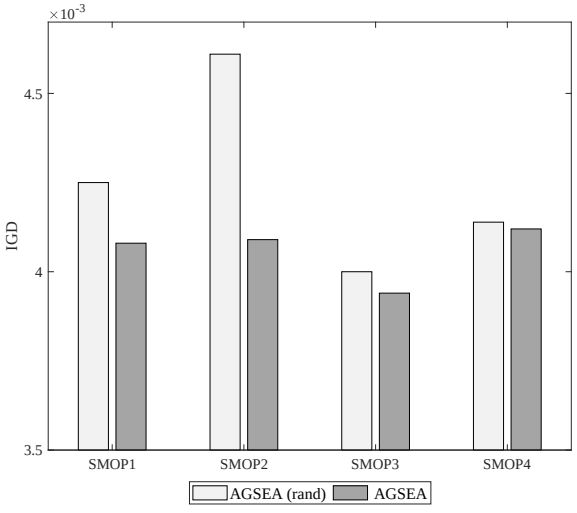

Figure 6: IGD values by AGSEA (rand) and AGSEA on SMOP1, SMOP2, SMOP3, and SMOP4 with 500 decision variables.

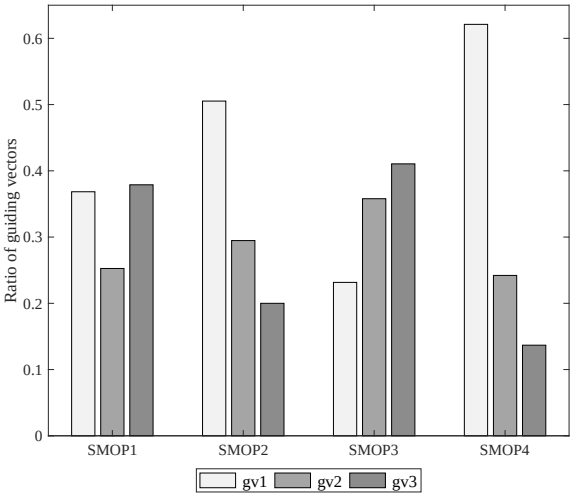

Figure 7: Ratio of guiding vectors selected by the proposed AGSEA during the optimization process of solving SMOP1, SMOP2, SMOP3, and SMOP4.

always select the same guiding vector but assembles multiple guiding vectors for different optimization problems, which can enhance the search ability to approximate the sparse Pareto solutions.

5. Conclusion

In this paper, we have proposed a deep reinforcement learning-based automated guiding vector selection method for large-scale sparse multi-objective optimization. By using deep neural networks to learn the mapping model between the population states and the expected cumulative rewards of guiding vectors, the proposed method can suggest promising guiding vectors to achieve better generalization performance. The selected guiding vector directs the clustering of decision variables and the development of novel variation operators, thus facilitating the generation of sparse solutions in high-dimensional decision spaces. The experimental results have demonstrated the superiority of the proposed algorithm over several state-of-the-art evolutionary algorithms.

In our future research, it is meaningful to investigate how to customize the guiding vectors based on the heuristic information for solving specific large-scale SMOPs (e.g., the feature selection problem [20]), which can enhance the performance of the proposed algorithm in real-world scenarios. In addition, since the proposed method only adopts the same variation operator for all the candidate guiding vectors, it is also desirable to use multi-agent reinforcement learning [18, 21] to simultaneously select different types of variation operators and guiding vectors to approximate sparse Pareto optimal solutions more efficiently.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 62136008, No.

62276001, No. U21A20512), in part by the Anhui Provincial Natural Science Foundation (No. 2308085J03, No. 2208085MF174), and in part by the Excellent Youth Foundation of Anhui Provincial Colleges (No. 2022AH030013).

References

[1] Agrawal, R., Srikant, R., et al., 1994. Fast algorithms for mining association rules, in: Proc. 20th int. conf. very large data bases, VLDB, Santiago, Chile. pp. 487–499.

[2] Bosman, P.A., Thierens, D., 2003. The balance between proximity and diversity in multiobjective evolutionary algorithms. IEEE transactions on evolutionary computation 7, 174–188.

[3] BÜYÜKKEÇECİ, M., OKUR, M.C., 2024. A comprehensive review of feature selection and feature selection stability in machine learning. Gazi University Journal of Science , 1–1.

[4] Chandra, N.K., Canale, A., Dunson, D.B., 2023. Escaping the curse of dimensionality in bayesian model-based clustering. Journal of Machine Learning Research 24, 1–42.

[5] Chatterjee, Y., Bourreau, E., Rančić, M.J., 2023. Solving various np-hard problems using exponentially fewer qubits on a quantum computer. arXiv preprint arXiv:2301.06978 .

[6] Christo, V.E., Nehemiah, H.K., Brighty, J., Kannan, A., 2022. Feature selection and instance selection from clinical datasets using co-operative co-evolution and classification using random forest. IETE Journal of Research 68, 2508–2521.

[7] Davuluri, S.K., Srivastava, D., Aeri, M., Arora, M., Keshta, I., Rivera, R., 2023. Support vector machine based multi-class classification for oriented instance selection, in: Proceedings of the 2023 International Conference on Inventive Computation Technologies, IEEE. pp. 112–117.

[8] Deb, K., Agrawal, R.B., et al., 1995. Simulated binary crossover for continuous search space. Complex systems 9, 115–148.

[9] Deb, K., Goyal, M., et al., 1996. A combined genetic adaptive search (geneas) for engineering design. Computer Science and informatics 26, 30–45.

[10] Derrac, J., García, S., Molina, D., Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1, 3–18.

[11] Ding, Z., Cao, L., Chen, L., Sun, D., Zhang, X., Tao, Z., 2023. Large-scale multimodal multiobjective evolutionary optimization based on hybrid hierarchical clustering. Knowledge-Based Systems 266, 110398.

[12] Ding, Z., Chen, L., Sun, D., Zhang, X., 2022. A multi-stage knowledge-guided evolutionary algorithm for large-scale sparse multi-objective optimization problems. Swarm and Evolutionary Computation 73, 101119.

[13] Fan, W., Ju, L., Tan, Z., Li, X., Zhang, A., Li, X., Wang, Y., 2023. Two-stage distributionally robust optimization model of integrated energy system group considering energy sharing and carbon transfer. Applied Energy 331, 120426.

[14] Feng, Y., Feng, L., Kwong, S., Tan, K.C., 2021. A multivariation multifactorial evolutionary algorithm for large-scale multiobjective optimization. IEEE Transactions on Evolutionary Computation 26, 248–262.

[15] Gao, M., Feng, X., Yu, H., Li, X., 2023. An efficient evolutionary algorithm based on deep reinforcement learning for large-scale sparse multiobjective optimization. Applied Intelligence , 1–24.

[16] He, C., Cheng, R., Yazdani, D., 2020. Adaptive offspring generation for evolutionary large-scale multiobjective optimization. IEEE Transactions on Systems, Man, and Cybernetics: Systems 52, 786–798.

[17] Huang, C., Zhou, X., Ran, X., Liu, Y., Deng, W., Deng, W., 2023. Co-evolutionary competitive swarm optimizer with three-phase for large-scale complex optimization problem. Information Sciences 619, 2–18.

[18] Huang, Y., Zhou, C., Cui, K., Lu, X., 2024. A multi-agent reinforcement learning framework for optimizing financial trading strategies based on timesnet. Expert Systems with Applications 237, 121502.

[19] Jiang, J., Han, F., Wang, J., Ling, Q., Han, H., Wang, Y., 2022. A two-stage evolutionary algorithm for large-scale sparse multiobjective optimization problems. Swarm and Evolutionary Computation 72, 101093.

[20] Karimi, F., Dowlatshahi, M.B., Hashemi, A., 2023. Semiaco: A semi-supervised feature selection based on ant colony optimization. Expert Systems with Applications 214, 119130.

[21] Kaven, L., Huke, P., Göppert, A., Schmitt, R.H., 2024. Multi agent reinforcement learning for online layout planning and scheduling in flexible assembly systems. Journal of Intelligent Manufacturing , 1–20.

[22] Kropp, I., Nejadhashemi, A.P., Deb, K., 2023. Improved evolutionary operators for sparse large-scale multiobjective optimization problems. IEEE Transactions on Evolutionary Computation .

[23] Lalou, M., Tahraoui, M.A., Kheddouci, H., 2018. The critical node detection problem in networks: A survey. Computer Science Review 28, 92–117.

[24] Lanthaler, S., Stuart, A.M., 2023. The curse of dimensionality in operator learning. arXiv preprint arXiv:2306.15924 .

[25] Li, K., Zhang, T., Wang, R., 2020. Deep reinforcement learning for multiobjective optimization. IEEE transactions on cybernetics 51, 3103–3114.

[26] Li, M., Zhen, L., Yao, X., 2017. How to read many-objective solution sets in parallel coordinates [educational forum]. IEEE Computational Intelligence Magazine 12, 88–100.

[27] Lin, Q., Li, J., Liu, S., Ma, L., Li, J., Chen, J., 2023. An adaptive two-stage evolutionary algorithm for large-scale continuous multi-objective optimization. Swarm and Evolutionary Computation 77, 101235.

[28] Liu, S., Lin, Q., Tian, Y., Tan, K.C., 2021. A variable importance-based differential evolution for large-scale multiobjective optimization. IEEE Transactions on Cybernetics 52, 13048–13062.

[29] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. nature 518, 529–533.

[30] Sangaiah, A.K., Javadpour, A., Ja?fari, F., Pinto, P., Zhang, W., Balasubramanian, S., 2023. A hybrid heuristics artificial intelligence feature selection for intrusion detection classifiers in cloud of things. Cluster Computing 26, 599–612.

[31] Shukla, S., 2023. Angle based critical nodes detection (abcnd) for reliable industrial wireless sensor networks. Wireless Personal Communications 130, 757–775.

[32] Si, L., Zhang, X., Tian, Y., Yang, S., Zhang, L., Jin, Y., 2023. Linear subspace surrogate modeling for large-scale expensive single/multi-objective optimization. IEEE Transactions on Evolutionary Computation .

[33] Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11, 341–359.

[34] Su, Y., Guo, N., Tian, Y., Zhang, X., 2020. A non-revisiting genetic algorithm based on a novel binary space partition tree. Information Sciences 512, 661–674.

[35] Tian, Y., Chen, H., Ma, H., Zhang, X., Tan, K.C., Jin, Y., 2022a. Integrating conjugate gradients into evolutionary algorithms for large-scale continuous multi-objective optimization. IEEE/CAA Journal of Automatica Sinica 9, 1801–1817.

[36] Tian, Y., Feng, Y., Zhang, X., Sun, C., 2022b. A fast clustering based evolutionary algorithm for super-large-scale sparse multi-objective optimization. IEEE/CAA Journal of Automatica Sinica 10, 1048–1063.

[37] Tian, Y., Li, X., Ma, H., Zhang, X., Tan, K.C., Jin, Y., 2022c. Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization. IEEE Transactions on Emerging Topics in Computational Intelligence .

[38] Tian, Y., Liu, R., Zhang, X., Ma, H., Tan, K.C., Jin, Y., 2020a. A multipopulation evolutionary algorithm for solving large-scale multimodal multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 25, 405–418.

[39] Tian, Y., Lu, C., Zhang, X., Cheng, F., Jin, Y., 2020b. A pattern mining-based evolutionary algorithm for large-scale sparse multiobjective optimization problems. IEEE Transactions on Cybernetics 52, 6784–6797.

[40] Tian, Y., Lu, C., Zhang, X., Tan, K.C., Jin, Y., 2020c. Solving large-scale multiobjective optimization problems with sparse optimal solutions via unsupervised neural networks. IEEE transactions on cybernetics 51, 3115–3128.

[41] Tian, Y., Shao, S., Xie, G., Zhang, X., 2023a. A multi-granularity clustering based evolutionary algorithm for large-scale sparse multiobjective optimization. Swarm and Evolutionary Computation , 101453.

[42] Tian, Y., Si, L., Zhang, X., Cheng, R., He, C., Tan, K.C., Jin, Y., 2021. Evolutionary large-scale multi-objective optimization: A survey. ACM Computing Surveys 54, 1–34.

[43] Tian, Y., Si, L., Zhang, X., Tan, K.C., Jin, Y., 2023b. Local model-based Pareto front estimation for multiobjective optimization. IEEE Transactions on Systems, Man, and Cybernetics: Systems 53, 623–634.

[44] Tian, Y., Zhang, X., Wang, C., Jin, Y., 2020d. An evolutionary algorithm for large-scale sparse multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 24, 380–393.

[45] Wang, L., Tian, Y., Xiang, X., Zhang, X., 2023. Optimizing large-scale distribution center locations during the covid-19 quarantine, in: 2023 IEEE Congress on Evolutionary Computation (CEC), IEEE. pp. 1–8.

[46] Wangsa, I.D., Vanany, I., Siswanto, N., 2023. An optimization model for fresh-food electronic commerce supply chain with carbon emissions and food waste. Journal of Industrial and Production Engineering 40, 1–21.

[47] Xiang, X., Tian, Y., Xiao, J., Zhang, X., 2020. A clustering-based surrogate-assisted multiobjective evolutionary algorithm for shelter location under uncertainty of road networks. IEEE Transactions on Industrial Informatics 16, 7544–7555.

[48] Zhang, Y., Tian, Y., Zhang, X., 2021. Improved sparseea for sparse large-scale multi-objective optimization problems. Complex & Intelligent Systems , 1–16.

[49] Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE transactions on Evolutionary Computation 3, 257–271.

[50] Zou, Y., Liu, Y., Zou, J., Yang, S., Zheng, J., 2023. An evolutionary algorithm based on dynamic sparse grouping for sparse large scale multiobjective optimization. Information Sciences 631, 449–467.

[51] Zuo, M., Gong, D., Wang, Y., Ye, X., Zeng, B., Meng, F., 2023. Process knowledge-guided autonomous evolutionary optimization for constrained multiobjective problems. IEEE Transactions on Evolutionary Computation .