Applied Intelligence (2023) 53:21116–21139
https://doi.org/10.1007/s10489-023-04574-9

An efficient evolutionary algorithm based on deep reinforcement learning for large-scale sparse multiobjective optimization

Mengqi Gao[1,2] · Xiang Feng[1,2] · Huiqun Yu[1,2] · Xiuquan Li[3]

Accepted: 12 March 2023 / Published online: 17 May 2023

Abstract

Large-scale sparse multiobjective optimization problems (SMOPs) widely exist in academic research and engineering applications. The curse of dimensionality and the fact that most decision variables take zero values make optimization very difficult. Sparse features are common to many practical complex problems currently, and using sparse features as a breakthrough point can enable many large-scale complex problems to be solved. We propose an efficient evolutionary algorithm based on deep reinforcement learning to solve large-scale SMOPs. Deep reinforcement learning networks are used for mining sparse variables to reduce the problem dimensionality, which is a challenge for large-scale multiobjective optimization. Then the three-way decision concept is used to optimize decision variables. The emphasis is on optimizing deterministic nonzero variables and continuously mining uncertain decision variables. Experimental results on sparse benchmark problems and real-world application problems show that the proposed algorithm performs well on SMOPs while being highly efficient.

Keywords Large-scale · Sparse multiobjective optimization · Evolutionary computation · Deep reinforcement learning

1 Introduction

Large-scale sparse multiobjective optimization problems are widely existing in the fields of evolutionary computation, cybernetics, and machine learning [1–3]. The neural network training in machine learning and the grid fault diagnosis problem in cybernetics both belong to SMOPs [4, 5]. Large-scale SMOPs contain numerous decision variables, and many objectives conflict with each other to some extent. Therefore, there is generally no single optimal solution for SMOPs. However, a set of Pareto-optimal solutions called the Pareto-optimal Set (PS) can be found [6]. For instance, the feature selection problem in classification refers to selecting the least number of features from numerous features to achieve optimal classification performance [7]. Its objectives of minimizing the chosen features and maximizing the correct rate may conflict. Another typical application is neural network weight training in deep learning, which aims to maximize classification accuracy while minimizing model complexity, where the improvement of one objective may lead to the deterioration of the other objectives [8]. Furthermore, large-scale SMOPs have sparse Pareto-optimal solutions, meaning that most decision variables take zero values. In a neural architecture search, there may be many useless links between neural networks and the optimal network structure to satisfy a specific need, which is sparse [9]. In feature selection problems, the selected relevant features are usually small relative to all features in the dataset [10, 11]. In power grid fault diagnosis, the faulty components that need to be identified quickly are very few compared to a large power system [12]. It is crucial to propose a generic black-box approach to address

✉ Xiang Feng
xfeng@ecust.edu.cn

Mengqi Gao
y10190116@mail.ecust.edu.cn

Huiqun Yu
yhq@ecust.edu.cn

Xiuquan Li
lixq@casted.org.cn

[1] Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China

[2] Shanghai Engineering Research Center of Smart Energy, Shanghai, China

[3] Chinese Academy of Science and Technology for Development, Beijing 100038, China

these large-scale sparse multiobjective optimization problems. Since most real-world SMOPs are conducted based on large datasets, they are also large-scale multiobjective optimization problems (LMOPs) [1].

In recent years, many efforts have been devoted to solving large-scale multiobjective optimization problems. The large-scale multiobjective evolutionary algorithms (MOEAs) used to solve LMOPs can be classified into decision variable grouping based [13], decision space reduction based [14], and novel search strategy based [15, 16]. The decision variable grouping based approach divides the decision variables into several groups and collaboratively optimizes the decision variables in each group. It generally includes random grouping, differential grouping, and decision variable analysis techniques [17–20]. The decision space reduction based methods usually use problem transformation or dimensionality reduction techniques to reduce the search space [14, 21]. Novel search methods mainly include novel reproduction operators and probabilistic models [15, 22]. For instance, the algorithm LMOCSO uses the search strategy of competing optimizers to improve efficiency [15].

Some of the above three classes of optimization algorithms can significantly reduce the search space and improve the performance of the algorithm when facing LMOPs [15, 20]. In contrast, their performance is limited when confronted with sparse multiobjective optimization problems. The methods based on decision variable analysis are prone to cause a high number of evaluations and thus waste time [20]. Methods based on problem transformation are prone to local optima [21], and novel search approaches may cause unstable convergence [15]. Most importantly, these algorithms are not designed for SMOPs, and they do not consider the sparsity of the problem, which leads to poor performance in solving SMOPs [23]. Tailoring MOEAs for sparse multiobjective optimization problems is urgently needed and necessary.

SparseEA is the first algorithm proposed for large-scale sparse multiobjective optimization problems. It designs a hybrid representation method, using a binary vector $mask$ and a real vector dec to compose the solution [24]. SparseEA optimizes decision variables based on the guidance of the prior information, but this prior knowledge is static. The evolution process may not fully utilize the sparse information of the mined non-dominated solutions [23]. MDR-SAEA uses a non-dominated sorting method to obtain prior knowledge and performs a multi-stage dimensionality reduction. Both algorithms are based on prior knowledge, but since prior knowledge is static and cannot be updated dynamically, they may not bring substantial performance improvement to the algorithm [23]. Another typical algorithm proposed for SMOPs by mining the sparse distribution of Pareto-optimal solutions to facilitate evolution. For example, the algorithm MOEAPSL suggests using unsupervised neural networks to learn Pareto-optimal subspaces to reduce the search space [14]. However, training the neural network by this algorithm is time-consuming and suffers from computational inefficiency [14]. In general, the research on large-scale SMOPs is still in its infancy, and there is an urgent need to propose an efficient method for solving SMOPs.

Considering that large-scale SMOPs are characterized by large-scale and sparse Pareto-optimal solutions, i.e., most decision variables are zero, we take mining nonzero variables as an entry point, reducing dimensionality by efficiently mining sparse Pareto-optimal solutions. Optimization of nonzero variables in the reduced search space significantly improves performance and efficiency. Therefore, we propose an efficient evolutionary algorithm based on deep reinforcement learning for large-scale SMOPs (MOEADRL). We introduce deep reinforcement networks to learn the sparsity of SMOPs in the academic research area. We also do experimental validation in solving practical application problems. The main contributions of this paper are summarized as follows.

(1) We propose an actor-critic-based sparse variable mining method for mining nonzero decision variables, which greatly reduces the search space. Unlike traditional deep reinforcement learning, we design an action space reduction method for sparse variable mining to overcome the limitations of deep reinforcement learning in solving optimization problems with large action spaces.
(2) We apply three-way decision theory to deal with uncertainty in the variable mining process. The deterministic variables are added to the all-zero or all-one group in the mining process, while the uncertain variables are momentarily placed in the mixed group. Maintaining an all-zero group can effectively reduce the search space and thus improve efficiency. The maintenance of a mixed group can reduce the sensitivity of grouping and further increase the grouping correctness, therefore mining sparse variables accurately.
(3) We design a resource allocation strategy for optimizing decision variables in different groups that reduces the need for computational resources. For the all-one group, we focus on the decision variables optimization; for the group mix, we concentrate on the sparse variables mining; and for the all-zero group, we do not make excessive resource allocation. Since most variables in sparse problems take zero value, we can save many computational resources by mining and maintaining zero variables.

The rest of this paper is organized as follows. In Section 2, the large-scale SMOPs are introduced, and existing MOEAs are reviewed. In Section 3, the proposed algorithm is described in detail. In Section 4, experimental results and analyses are given. Finally, conclusions are drawn, and future work is outlined in Section 5.

2 Background

In this section, we give the definition of the large-scale SMOPs, review the MOEAs for large-scale SMOPs, and finally introduce the basic concepts and applications of actor-critic deep reinforcement learning methods.

2.1 Large-scale sparse multiobjective optimization

A multiobjective optimization problem (MOP) can be mathematically defined as

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

where $\mathbf{x} = (x_1, \ldots, x_D) \in \Omega$ is the decision vector in the feasible region, $\Omega \subseteq R^D$ is the feasible decision space, $\mathrm{f} : \Omega \rightarrow \Lambda \subseteq \mathbb{R}^M$ consists of M objectives, and Λ is the objective space.

The MOP involves more than 100 decision variables can be called an LMOP. LMOPs can contain hundreds or thousands of decision variables, some of which may take zero values. The LMOPs with most decision variables taking zero values are called large-scale SMOPs.

The SMOPs can be found in a variety of real-world applications, including pattern mining [25], feature selection [26], crucial node detection [27], and so on. Although some algorithms have been proposed to solve the above sparse problems, these algorithms are limited to specific objective functions. Offering algorithms to solve different large-scale SMOPs in a black-box manner has theoretical and practical implications [14].

Traditional MOEAs may require numerous evaluations to obtain the Pareto-optimal solutions because the decision space expands exponentially with the increase of decision variables. The number of nonzero decision variables in the Pareto-optimal solution of the SMOPs is much smaller than the total number of decision variables. As a result, utilizing traditional MOEAs to deal with large-scale SMOPs is prohibitively expensive and may not be optimized for nonzero decision variables [14, 28].

Tian et al. proposed the algorithm SparseEA to solve SMOPs [24]. SparseEA designs a new population initialization strategy that uses a hybrid representation of the solution to integrate two different encodings. More specifically, the solution in SparseEA consists of two components, a real vector dec representing the decision variables' value and a binary vector $mask$ representing the mask [24]. MOEAPSL solves large-scale SMOPs by approximating the Pareto-optimal subspace and subsequently performs genetic operations in the learned Pareto-optimal subspace to generate the offspring solution [14]. This method inherits the population initialization approach of SparseEA and uses an unsupervised neural network to learn the sparse distribution and compact representation of the decision variables. The two methods mentioned above proved the effectiveness of the hybrid representation of solutions in solving SMOPs. Zhang et al. state that the hybrid representation strategy should not only focus on sparse variable detection and maintenance but also on nonzero variables being fully optimized within a finite function evaluation budget [29]. They suggested an improved SparseEA for large-scale SMOPs, namely, SparseEA2. Inspired by the above algorithms, we will use a hybrid representation of the solution for population initialization and perform dec optimization while focusing on the sparse decision variable mining. We tend to identify the proper nonzero decision variables by optimizing $mask$ while optimizing dec to find the optimal value of each nonzero decision variable. In this instance, rather than solving the MOPs for D variables, we only need to do it for nonzero variables.

Finding nonzero decision variables and performing optimization of dec is the key to reducing the dimensionality of large-scale SMOPs. It is crucial to locate nonzero decision variables appropriately and efficiently. We attempt to use the actor-critic method to mine sparse variables, which was inspired by the methods MOEAPSL and DRL-EC3 [14, 30]. The actor-critic method is proven to be more efficient and more advanced in performing mining [31]. We will then introduce some fundamental concepts and the applications of actor-critic in research.

2.2 Actor-critic deep reinforcement learning

Actor-critic is an emerging deep reinforcement learning method with fast convergence and the ability to handle large action spaces [32]. Actor-critic deep reinforcement learning combines value-based, and policy-based approaches in reinforcement learning [33]. The actor is based on a policy gradient, and the policy is parameterized as a neural network. The actor is in charge of outputting the policy and selecting the actions. The critic is responsible for calculating the score for each action [34]. The critic observes the state and gives a score to the actor. If the critic assigns a poor score to an action, the probability of outputting that action is reduced.

Yuan et al. proposed an actor-critic-based deep stochastic online scheduling algorithm to handle the energy

minimization problem in UAV-assisted networks [33]. The algorithm enables a workable, fast convergence and dynamically adaptive solution. Wei et al. deploy an actor-critic reinforcement learning framework to tackle the joint decision problem to reduce the average end-to-end delay in fog-enabled IoT due to the stochastic nature of wireless signals and service requests [34]. Liu et al. propose an actor-critic architecture with deep reinforcement learning to cope with the classical job shop problem. Moreover, experimentally demonstrated that the quality of the solutions found by the model is also better [35]. Kyriakos et al. proposed an online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem, and simulation examples showed the effectiveness and stability of the algorithm [36].

Although actor-critic deep reinforcement learning has been used for various applications and some control optimization problems [33, 36], it has yet to be used before to address the challenge of mining sparse variables in large-scale SMOPs. We will employ actor-critic to mine nonzero variables due to its advantages of tolerating uncertainty, converging quickly, and handling large-scale spaces [32, 37]. In the following section, we will introduce the specifics of the proposed algorithm.

3 Proposed algorithm

First, we will give the basic framework of the proposed algorithm. Then, the sparse variable mining method based on deep reinforcement learning is introduced. Finally, we offer a method for grouping and optimizing decision variables based on the concept of the three-way decision.

3.1 Framework

The general framework of the proposed algorithm is shown in Fig. 1. An actor-critic method is used to mine the decision variables whose $mask$ takes the value of 1. All decision variables are divided into three groups based on the concept of three-way decision: G_{zero}(all-zero), G_{one}(all-one), G_{mix}(mix). The significance of grouping is to find sparse variables and optimize the dec of sparse variables. Referring to the method shown in Fig. 2, we obtain the population by multiplying the dec and $mask$ pointwise. Then, using the tournament selection method to choose $parent1$, $parent2$, and $parent3$ for offspring generation. Finally, based on the three-way decision concept, different strategies are used in the crossover mutation of the deterministic groups G_{zero}, G_{one}, and the uncertainty group G_{mix}, respectively.

The algorithm flow is shown in Algorithm 1. First, initializing the population P and performing non-dominant sorted in population individuals. Then the all-zero variable set G_{zero}, the all-one variable set G_{one}, and the mixed set G_{mix} are set to null. Iterative optimization is performed on the population until the stopping condition is satisfied. Specifically, nonzero variables are mined using deep reinforcement networks to obtain masks. We apply the three-way decision concept to group decision variables. Then, the parents are selected using the tournament selection method. Finally, the decision variables in different groups are crossover-mutation in various ways, and environment selection is performed.

3.2 Sparse variable mining

3.2.1 Overview of actor-critic-based sparse variable mining

The solution is represented in a hybrid representation using two different encodings, binary vector $mask$ and real vector dec [24]. Each solution $\mathbf{x}$ is obtained by

Input: N (Population size), D (decision variables size), $maxstep$ (maximum number of steps per episode)
Output: Population (the final population)
1: $P(dec.*mask) \leftarrow$ Initial population;
2: NDS(non-dominated solution) $\leftarrow$ Non-dominant sorting(P);
3: $G_{\text{zero}} \leftarrow \emptyset$, $G_{\text{one}} \leftarrow \emptyset$, $G_{\text{mix}} \leftarrow \emptyset$;
4: $\rho \leftarrow 0.7$; // ratio of offspring solutions generated according to the three-way decision
5: **while** termination condition not reached **do**
6: $mask \leftarrow$ Sparse variable mining($NDS, maxstep$);
7: $[G_{\text{zero}}, G_{\text{one}}, G_{\text{mix}}] \leftarrow$ Three-way decision-based variable grouping($mask, D$);
8: $[parent1, parent2, parent3] \leftarrow$ Select parents based on the tournament selection;
9: $P \leftarrow$ Variation and environmental selection($parent1, parent2, parent3, G_{\text{zero}}, G_{\text{one}}, G_{\text{mix}}, \rho, P$);
10: **end while**
11: **return** P

Algorithm 1 Main framework.

$$\mathbf{x} = (\text{mask}_1 \times \text{dec}_1, \cdots, \text{mask}_D \times \text{dec}_D), \tag{2}$$

here, D specifies the number of decision variables as well as the dimensionality of the optimization problem. dec represents the value of the decision variable, and $mask$ indicates whether the decision variable is zero.

The values of the individual decision variables are obtained by the product of $mask$ and dec. If the decision variable $mask$ takes value 0, that decision variable takes

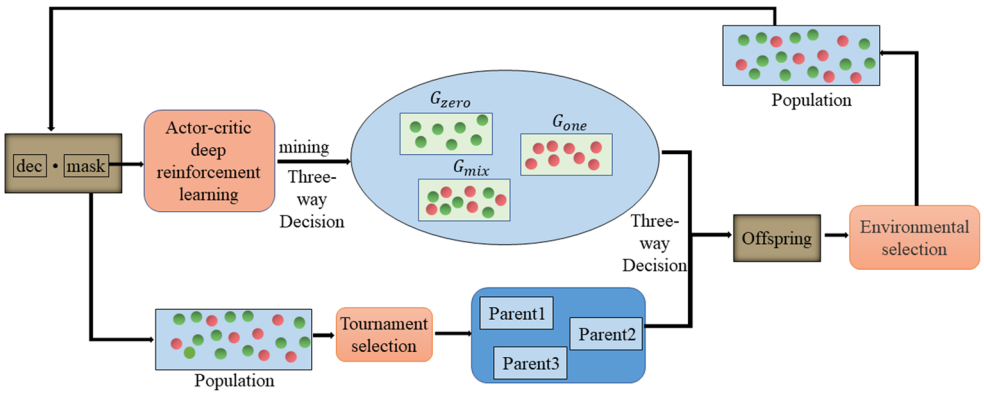


Fig. 1 Framework of the proposed method

	mask						
Individual 1	0	1	1	1	0	1	0
Individual 2	1	0	1	0	0	1	1
Individual 3	0	1	0	0	0	1	1
Individual 4	0	0	0	1	1	0	1

•

dec						
0.4	0.3	0.7	0.7	0.2	0.5	0.4
0.2	0.7	0.8	0.5	0.9	0.1	0.4
0.3	0.2	0.9	0.5	0.7	0.8	0.5
0.6	0.5	0.3	0.8	0.9	0.6	0.4

=

population:mask.* dec						
0	0.3	0.7	0.7	0	0.5	0
0.2	0	0.8	0	0	0.1	0.4
0	0.2	0	0	0	0.8	0.5
0	0	0	0.8	0.9	0	0.4

Fig. 2 Binary vector *mask* and real vector *dec* hybrid representation solution

value 0. Figure 2 shows the *dec* and *mask* of four individuals and the values of corresponding variables. Only decision variables with a *mask* of 1 can have a nonzero value. If the *mask* is 0, the final value of the decision variable is 0, regardless of the value of the *dec*.

Most of the decision variables in SMOPs are zero, so it would be wasteful to perform the same optimization for each decision variable [24]. The detection and optimization of nonzero decision variables would be exceedingly efficient. Inspired by Chen et al. suggested rule mining through reinforcement learning, we use deep reinforcement learning for sparse variables mining [31]. The deep actor-critic can handle large action spaces and complex state spaces well with fast convergence properties [30, 33, 38]. So, the large-scale SMOPs with a large decision space are suitable to be solved by actor-critic deep reinforcement learning methods.

The actor-critic used in this paper divides the agent into two parts, an actor and a critic. The actor is based on the policy gradient, which can select the appropriate action in the continuous action space. The actor network acts as a policy network, which takes the observation as input and returns the action that maximizes the expected cumulative long-term reward as output, thus implementing strategy. π is employed to denote the strategy of the actor, and $\pi(A \mid S;\theta)$ is used to output the conditional probability that an actor with parameter θ chooses action A in state S. The critic is value-based and is used to evaluate the decision of the actor by Q value. Critic acts as a value network that maps observations to scalar values. This scalar value represents the expected total long-term reward that the agent is expected to accumulate when starting from a given observation and performing a given action. The corresponding expectation of the critic with parameter ϕ to accept observation S and return the discounted long-term reward is denoted by the symbol $V(S;\phi)$. The value function critic only needs observation as input.

During training, the actor-critic agent: estimates the probability of taking each action in the action space and randomly selects actions based on the probability distribution, using the current policy to interact with the environment in numerous steps before updating actor and critic attributes.

The actor and critic agents are initialized using random parameter values θ and ϕ, respectively. The actor generates T experiences based on the current policy in each optimization episode. The episode experience sequence is

$$S_{ts}, A_{ts}, R_{ts+1}, S_{ts+1}, \ldots, S_{ts+T-1}, A_{ts+T-1}, R_{ts+T}, S_{ts+T}, \quad (3)$$

here, ts is the starting step of the current T experiences. At the beginning of the training episode, $ts = 1$. S_{ts} is the state observation of the ts experience, and A_{ts} is the action taken at state S_{ts}. S_{ts+1} is the next state, and R_{ts+1} is the reward received for transferring from state S_{ts} to state S_{ts+1}.

For each experience $t = ts + 1, ts + 2, \ldots, ts + T$, calculate the return G_t, which is the sum of the reward for that step and the discounted future reward. If S_{ts+T} is not

a terminal state, the discounted future reward includes the discounted state value function, calculated using the critic network V,

$$G_t = \sum_{k=t}^{ts+T} \left(\gamma^{k-t} R_k\right) + b\gamma^{T-t+1} V\left(S_{ts+T} \mid \phi\right), \tag{4}$$

here, b is 0 if S_{ts+T} is a terminal state and 1 otherwise. γ is a discount factor, and $0<\gamma<1$. The discount factor specifies the time horizon that the future reward should be considered. Agents will learn to act only for short-term rewards if γ is small. If γ is large, it can facilitate agents learning to work for long-term rewards, but agents must receive some timely rewards, so γ cannot be set to 1. The discount factor γ is specified.

The advantage function D_t is then computed using the return G_t and the critic network V,

$$D_t = G_t - V\left(S_t \mid \phi\right), \tag{5}$$

the advantage function shows the improvement in the expected cumulative reward gained when taking action in the state S_t, compared to the average of all possible actions taken in the state S_t [39]. The advantage function can improve learning efficiency while making learning more stable; in addition, the advantage function also helps reduce variance, which is a significant factor leading to overfitting [40, 41].

To maximize the expected discounted reward, accumulate the gradients for the actor network by following the policy gradient [42]. The gradient of the actor network can be calculated by

$$d\theta = \sum_{t=1}^{T} \nabla_{\theta_\mu} \ln \pi\left(S_t \mid \theta\right) \cdot D_t. \tag{6}$$

The gradient descent method is used to update the parameter vector θ of the actor network,

$$\theta = \theta + \delta d\theta, \tag{7}$$

here, δ is the learning rate of the actor. δ is vital to maximizing the speed of learning. A small learning rate causes slow learning, while a large learning rate induces oscillations. We specify the learning rate when creating the actor.

The actor is strategy-based, and the critic computes the value of the corresponding action to provide feedback to the actor, primarily evaluating the strategy of the actor. Across all T experiences, accumulate the gradients for the critic network by minimizing the mean squared error loss between the estimated value function $V(t)$ and the computed target return $G(t)$.

$$d\phi = \sum_{t=1}^{T} \nabla_\phi \left(G_t - V\left(S_t \mid \phi\right)\right)^2. \tag{8}$$

The parameter ϕ of the critic network is also updated according to the gradient descent method,

$$\phi = \phi + \omega d\phi, \tag{9}$$

here, ω is the learning rate of the critic. We specify the learning rate when creating the critic.

3.2.2 Problem reformulation

The sparse variable mining problem is formulated as a Markov decision process (MDP). Following are definitions of states, actions, and rewards.

(1) States. The system state HV evaluates the population, and this evaluation metric can estimate both the distributivity and convergence of the population [43]. The state HV of the system is determined by the population mask in the system, while the population mask of the following state is determined only by the population of the current state and the action taken by the system, so the transition of the state is in accordance with the MDP.

HV is defined as the hypervolume of the region enclosed by the points in population P and the points in reference point set PF^*. The larger HV indicates better convergence and distribution of P [43]. HV indicator can be defined as

$$HV\left(P, PF^*\right) = \lambda\left(H\left(P, PF^*\right)\right), \tag{10}$$

where λ denotes the Lebesgue measure, and $H(P, R) = \{\mathbf{z} \in Z \mid \exists \mathbf{p} \in P, \exists \mathbf{r} \in R : \mathbf{p} \leq \mathbf{z} \leq \mathbf{r}\}$. The set of reference points in HV is the reference points away from the Pareto front.

(2) Actions. The value of the sparse variable *mask* is the system action. The system takes action to assign a value of 0 or 1 to the *mask* of a population individual. The purpose is to mine variables with *mask* of 1, which are sparse variables. Each individual's *mask* taking value 0 or 1 is a discrete action, and the action space may be huge as the decision variables of the optimization problem increase [44]. We take the approach of continuousizing the actions by setting them in the interval [0, 1] and then mapping the output to 0 or 1 [45]. This approach can reduce the action space and overcome the limitations of deep reinforcement learning in solving optimization problems with large action spaces.

(3) Rewards. Rewards are used to evaluate the impact of actions on the environment [46]. The goal of reinforcement learning is to obtain maximum cumulative rewards in the long-term [47]. A well-designed reward can help to take the optimal strategy and guide the learning process, which can guide the population evolution to obtain a population with convergence and diversity. Therefore, we design a reward function that rewards or penalizes the action by evaluating the population state. Specifically, at state S_t, the agent takes action A_t; the system state is transferred to the next state S_{t+1}, and the agent gets reward R [48]. In other words, at each iteration of the optimization process, the deep reinforcement learning network takes action and outputs the value of the $mask$. After performing that action, the system gives a reward or penalty. Positive rewards imply that the actions taken resulted in a population with better diversity and convergence, while negative rewards indicate a deterioration in the population quality. Then the reward function can be set as,

$$R = \begin{cases} +R_{pos} \text{ (positive reward)}, & S_{t+1} \text{ is better than } S_t \\ -R_{neg} \text{ (negative reward)}, & S_t \text{ is better than } S_{t+1} \end{cases} \tag{11}$$

where $+R_{pos}$ denotes positive rewards and $-R_{neg}$ denotes negative rewards. The population evaluation criterion HV represents the system state. S_{t+1} represents the HV for experience $t+1$, and S_t represents the HV for experience t. Comparing the HV of states S_{t+1} and S_t, if HV at S_{t+1} are better than S_t, give a positive reward $+R_{pos} = 1$; if HV at S_{t+1} are worse than S_t, give a negative reward $-R_{neg} = -1$ [49]. In this case, the reward is available to the agent at most states rather than only when the final goal is reached. Using such a relatively dense reward setting allows the agent to distinguish the good and bad states quickly [50].

3.2.3 Sparse variable mining algorithm

The sparse variable mining algorithm is shown in Algorithm 2, and the actor-critic framework is shown in Fig. 3. First, we load the environmental observations to obtain the metric HV that evaluates the population. Based on these metrics, we decide whether the action should be rewarded. For training purposes, actor and critic agents are created. The actor is accountable for taking actions based on observations, and the critic provides supervisory feedback to the actor. Next, buffers are initialized to store experience data: observation, action, and reward caches. Each episode collects observation, action, and reward data, which is then used to update actor and critic parameters. Following the start of training, the actor outputs an action for current obs, and the function Step is used to compute the reward for this action. As shown in Algorithm 3, depending on the current observation and the action taken, the Step function can output the next state, the reward for the action, and the updated population $mask$. The current action, observation, and reward are stored in the buffer (lines 5-7). Repeat steps 5-7 until $maxstep$ is satisfied. Following that, the training data is collected as a batch of experiences, and the episode rewards are calculated, which are discounted future rewards. Finally, the gradient of the loss function concerning the policy representation parameters is computed and used to update the actor and critic network parameters. The gradient of actor and critic is calculated as (6) and (8). The actor and critic networks are updated according to (7) and (9).

Input: $Mask$ (the mask of the NDS), $maxstep$ (maximum number of steps per episode)
Output: $Mask$ (the mask of the population)
1: $obs \leftarrow$ Observe environment(HV);
2: [actor, critic] $\leftarrow$ Create actor-critic agent;
3: [observationBuffer, actionBuffer, rewardBuffer] $\leftarrow$ Create buffers to store experiences;
4: **for** $i = 1 : maxstep$ **do**
5: action $\leftarrow$ GetAction(actor, obs);
6: [nextObs, reward, $Mask$] $\leftarrow$ Step(obs, action, $Mask$);
7: [observationBuffer, actionBuffer, rewardBuffer] $\leftarrow$ Update the action, observation, and reward experiences;
8: **end for**
9: [observationBatch, actionBatch, rewardBatch] $\leftarrow$ Create training data from the buffer;
10: discountedReturn $\leftarrow$ Compute the discounted future reward;
11: lossData $\leftarrow$ Organize data to pass to the loss function;
12: gradient $\leftarrow$ Compute the gradient of the loss;
13: [actor, critic] $\leftarrow$ Update the actor and critic network using the computed gradients;
14: **return** $Mask$

Algorithm 2 Sparse variable mining.

3.3 Three-way decision-based variable grouping method

We have completed training the actor-critic network and obtained the values of $mask$. The significance of dividing the $mask$ into 0 and 1 is to optimize the sparse variable dec. Then it is critical to find the decision variables of $mask$ takes 1. Once the value of the mask is confused, it is worse

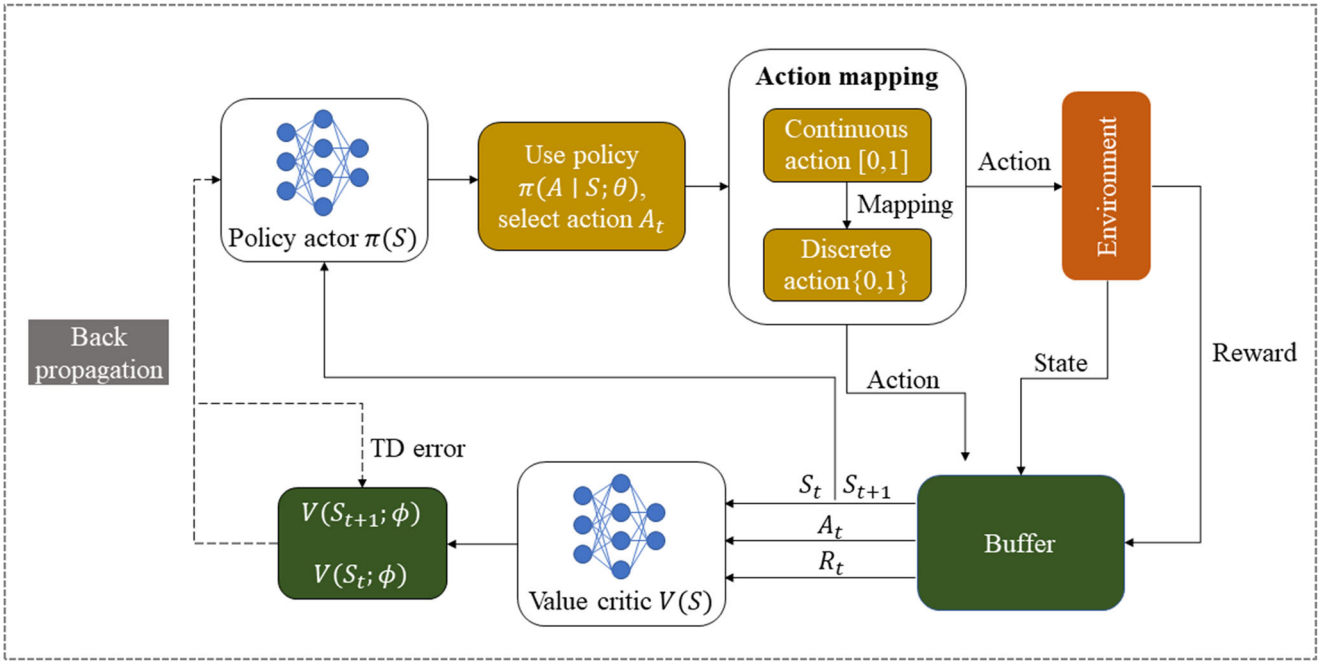


Fig. 3 The actor-critic framework for MOEADRL

Input: obs (HV), action, $Mask$
Output: nextObs, reward, $Mask$

1: $Mask \leftarrow$ action; // take action to update the population mask
2: NextObs($NeHV$) $\leftarrow$ Observe the state after taking action;
3: **if** $NeHV > HV$ **then**
reward = 1;
4: **else if** $NeHV < HV$ **then**
reward = -1;
5: **end if**
6: **return** NextObs, reward, $Mask$

Algorithm 3 Step function.

than not optimizing the decision variable because blindly optimizing zero variables also wastes evaluation. Therefore, it is counterproductive to assign uncertain decision variables to all-one or all-zero groups directly.

Decision variables known with a high degree of certainty can be directly added to the all-one or all-zero groups. And for decision variables that are not sufficiently understood, they are temporarily added to the mixed group and wait for subsequent mining before making decisions. This corresponds to the concept of three-way decision-making, where in many practical situations, the available knowledge is often incomplete and imprecise [51]. In such cases, it may not be possible to determine the definite membership of the subject [52]. Therefore, the actual decision problem can be solved by accepting, rejecting, or delaying the decision. Delaying decisions is a better and more practical strategy to lessen the impact of uncertainty in decision-making when there is insufficient information [52, 53]. People's decisions can be made gradually as information is updated and supplemented [54].

Following the concept of the three-way decision, two-state sets and three action sets can be used to describe the decision-making process for the grouping of decision variables [51]. The state set StS = {Z, O} contains two complementary relations of states Z and $\neg Z$(O), indicating that the object belongs to Z and not belonging to Z. In sparse variable mining algorithms, it means that the decision variables are either zero variables or not. The action set AtS = {aZ, aM, aO} denote the receiving decision, the delaying decision, and the rejecting decision, respectively. When z belongs to Z, $\lambda\ ZZ$, λMZ, and λOZ denote the losses under taking actions aZ, aM, and aO, respectively. λZO, λMO, and λOO denote the losses of taking actions aZ, aM, and aO when z does not belong to Z, respectively. The loss function of the three-way decision is shown in Table 1.

Therefore, the expected loss cost of object z is as,

$$\begin{cases} R\left(aZ \mid [z]_R\right) = \lambda_{ZZ} \Pr\left(Z \mid [z]_R\right) + \lambda_{ZO} \Pr\left(O \mid [z]_R\right) \\ R\left(aM \mid [z]_R\right) = \lambda_{MZ} \Pr\left(Z \mid [z]_R\right) + \lambda_{MO} \Pr\left(O \mid [z]_R\right) \\ R\left(aO \mid [z]_R\right) = \lambda_{OZ} \Pr\left(Z \mid [z]_R\right) + \lambda_{OO} \Pr\left(O \mid [z]_R\right) \end{cases} \tag{12}$$

where $\Pr\left(Z \mid [z]_R\right)$ is a conditional probability function. Without loss of generality, we assume that $\lambda ZZ \leqslant \lambda MZ$

Table 1 Loss function of the three-way decision

Action	Loss function	
	Z	O
aZ	λZZ	λZO
aM	λMZ	λMO
aO	λOZ	λOO

$\leqslant \lambda OZ$ and $\lambda ZO \leqslant \lambda MO \leqslant \lambda OO$, and according to the Bayesian minimum risk decision [55, 56], we have

$$\begin{cases} \text{(Z) If } R\left(aZ \mid [z]_R\right) \leqslant R\left(aM \mid [z]_R\right) \text{ and} \\ \quad R\left(aZ \mid [z]_R\right) \leqslant R\left(aO \mid [z]_R\right), \text{ then } z \in \text{Zero}(Z) \\ \text{(M) If } R\left(aM \mid [z]_R\right) \leqslant R\left(aZ \mid [z]_R\right) \text{ and} \\ \quad R\left(aM \mid [z]_R\right) \leqslant R\left(aO \mid [z]_R\right), \text{ then } z \in \text{Mix}(Z) \\ \text{(O) If } R\left(aO \mid [z]_R\right) \leqslant R\left(aZ \mid [z]_R\right) \text{ and} \\ \quad R\left(aO \mid [z]_R\right) \leqslant R\left(aM \mid [z]_R\right), \text{ then } z \in \text{One}(Z) \end{cases} \tag{13}$$

here, $Zero(Z)$, $Mix(Z)$, and $One(Z)$ represent dividing the object z into the positive, boundary, and negative domains of Z, respectively. In the problem of mining sparse variables in this paper, it means dividing the variables into G_{zero}, G_{one}, and G_{mix} groups.

According to (12) and (13), we can obtain

$$\begin{cases} \text{(Z1) If } \Pr\left(Z \mid [z]_R\right) \geqslant \alpha, \text{ then } z \in \text{Zero}(Z) \\ \text{(M1) If } \beta < \Pr\left(Z \mid [z]_R\right) < \alpha, \text{ then } z \in \text{Mix}(Z) \\ \text{(O1) If } \Pr\left(Z \mid [z]_R\right) \leqslant \beta, \text{ then } z \in \text{One}(Z) \end{cases} \tag{14}$$

where

$$\alpha = \frac{\lambda_{ZO} - \lambda_{MO}}{(\lambda_{ZO} - \lambda_{MO}) + (\lambda_{MZ} - \lambda_{ZZ})}$$

$$\beta = \frac{\lambda_{MO} - \lambda_{OO}}{(\lambda_{MO} - \lambda_{OO}) + (\lambda_{OZ} - \lambda_{MZ})}.$$

The values of α and β will be discussed in detail in the experimental section. We will group the decision variables using (14) and add them to the G_{zero}, G_{one}, or G_{mix} groups. The algorithm flow is shown in Algorithm 4.

Input: $mask$ (the mask of the population), D (decision variables size)
Output: G_{zero}, G_{one}, G_{mix}

for i =1: D **do**
  count(mask(i)) ← Calculate the number of times the decision variable i takes 0 in the mask matrix;
  if count(mask(i))/D ≥ α **then**
    G_{zero} ← mask(i);
  else if count(mask(i))/D ≤ β **then**
    G_{one} ← mask(i);
  else
    G_{mix} ← mask(i);
  end if
end for
return G_{zero}, G_{one}, G_{mix}

Algorithm 4 Three-way decision-based variable grouping.

Input: $p1$ ($mask.*dec$), $p2$ ($mask*dec$), $p3$ ($mask.*dec$), G_{one} (the decision variables are all one), G_{zero} (the decision variables are all zero), G_{mix} (uncertain decision variables), ρ (ratio of offspring generated according to the three-way decision), P
Output: P_2 (offspring population)

// Optimization of real vectors;
OffDec(G_{one}) ← OperatorDE($p1dec(G_{\text{one}})$, $p2dec(G_{\text{one}})$, $p3dec(G_{\text{one}})$);
// Optimization of binary vectors
OffMask ← false;
if $\rho >$ rand() **then**
  OffMask(G_{one}) ← True;
  OffMask(G_{mix})←BinaryOperator($p1mask(G_{\text{mix}})$, $p2mask(G_{\text{mix}})$);
else
  OffMask ← BinaryOperator ($p1mask$, $p2mask$);
end if
P_1 ← OffMask.∗ OffDec; // generate offspring
P_2 ← Environmentalselection(P, P_1);
return P_2

Algorithm 5 Variation and environmental selection.

3.4 Variation and environmental selection

We have completed grouping decision variables in the previous section. The following focus is on optimizing the variable dec in the identified G_{one} and the further mining of nonzero variables. The algorithm flow is shown in Algorithm 5.

Firstly, the real vector dec is optimized, and the differential evolution operator is applied to generate offspring [57]. It is worth noting that we only optimize the dec of nonzero decision variables. This not only enables targeted optimization of decision variables but also saves resources. The optimization of the binary vector $mask$ is then performed. We set the balance factor ρ to determine the way to generate offspring masks [14]. When ρ exceeds $rand$, the offspring inherit the masks of the decision variables in G_{one} and G_{zero}. Only the masks of the decision variables in G_{mix} are performed crossover-mutation. When ρ is less than or equal to $rand$, it directly inherits parent1's $mask$ and performs crossover mutation to generate the offspring $mask$. The algorithm 6 illustrates the crossover mutation of the mask. The mask of the first parent is given to the offspring, and then the offspring mask flip is carried out.

The binary vector $mask$ flipping probability is calculated by (15) [1].

$$\begin{aligned} p_1 &= \frac{pd}{2n} \\ p_0 &= \frac{pd}{2(d-n)}, \end{aligned} \tag{15}$$

The probability of performing mutation is p. Suppose that the solution to be mutated contains d decision variables, where n nonzero decision variables, $d - n$ zero decision variables. p_1 denotes the probability of flipping a nonzero variable and p_0 denotes the probability of flipping a zero variable.

The final step was to select individuals from the population using the well-known environmental selection method [58]. During environmental selection, all non-dominated solutions are first selected as the next generation population P_2, and environmental selection ends when the number of individuals in population P_2 is N. When the number of individuals in population P_2 is not equal to N:

(1) If the number of individuals $|P_2| < N$, the best $N - |P_2|$ dominant solutions are selected and added to population P_2.
(2) If the number of individuals $|P_2| > N$, apply the truncation method to delete the redundant non-dominated solutions in P_2 until $|P_2|$ equals to N. The truncation method means deleting non-dominated solutions closest to the neighboring solutions. This truncation method prevents the boundary solutions from being deleted.

Other environment selection methods can also be used in our algorithm MOEADRL [24], given that environment selection is independent of the core components of the algorithm, decision variable mining, and grouping. In the following section, we will conduct experiments to demonstrate the algorithm's effectiveness.

4 Experimental studies

In this section, we first give experimental results and analysis of the algorithm parameters and then verify the effectiveness of the proposed algorithm. We compare the performance of the proposed algorithm with five representative algorithms, comparing and analyzing the result on benchmark problems, algorithm convergence, population distributivity, algorithm computational efficiency, and practical applications. Testing was conducted on a personal computer with Intel(R) Core(TM) i7, 2.30 GHz CPU, 8 GB memory, and Windows 10. The algorithm is implemented using Matlab-R2020b.

Input: $p1mask$, $p2mask$
Output: OffMask
1: OffMask ← $p1mask$;
2: **for** i = 1 : length($p2mask$) **do**
3: dis ← xor($p1$ mask(i), $p2$ mask(i)); // different variables in $p1mask$ and $p2mask$
4: pro_{dis} ← Calculate the probability of flipping each variable in dis OffMask(i);
5: Flip each variable in dimensions dis of OffMask(i) with the probability pro_{dis};
6: pro_{flip} ← Calculate the probability of flipping each variable in dis of OffMask(i);
7: Flip each variable of OffMask(i) with the probability pro_{flip};
8: **end for**
9: **return** OffMask

Algorithm 6 BinaryOperator.

4.1 Test problems and comparison algorithms

4.1.1 Test Problems

We will verify the performance of the proposed algorithm on the sparse multiobjective benchmark set SMOP, which contains eight benchmark functions SMOP1-SMOP8 [24]. We set the objective to two and three and the decision variables to 300, 500, 800, and 1000 in the experiments. The sparsity of the Pareto-optimal solution in the numerical experiments is 0.1 [24].

In addition, we will test the performance of the proposed algorithm in three practical applications. We apply the proposed algorithm to the neural network training problem NN, portfolio optimization problem PO, and sparse signal reconstruction problem SR [14].

4.1.2 Comparison algorithms

Five state-of-the-art MOEAs were compared with the proposed algorithm, including two excellent MOEAs, MOEADVA and LMEA for LMOPs, and three customized MOEAs for SMOPs, SparseEA, MOEAPSL, and SparseEA2. MOEADVA is a large-scale multiobjective evolutionary algorithm based on decision variable analysis, classifying decision variables into the distance, location, and mixed variables [20]. This method improves the convergence speed and diversity of MOEAs. LMEA uses a divide-and-conquer strategy to group decision variables and co-optimizing them [13]. SparseEA is the first MOEA proposed for SMOPs, which designs a new population initialization strategy and a genetic operator to optimize

Table 2 Algorithm parameter settings

Algorithms	Parameter settings
MOEADVA	NCA=20, solutions in control variable analysis; CR=1; mu=1/D.
LMEA	$nSel$=5, solutions for decision variable clustering; $nPer$=50, number of perturbations on each solution; $nCor$=5, solutions for variable interaction analysis.
SparseEA	CR=1; mu=1/D.
MOEAPSL	ρ=0.5, the rate of generating offspring solutions in the Pareto-optimal subspace; CR=1; mu=1/D.
SparseEA2	Group=4; CR=1; mu=1/D.
	R=1, -1; CR=1; mu=1/D.

real vectors [24]. MOEAPSL is a customed MOEA for SMOPs that uses unsupervised neural networks to learn Pareto-optimal subspaces, thus reducing the search space [14]. SparseEA2 is an improvement for SparseEA, which improves the efficiency of generating Pareto-optimal solutions by connecting dec and $mask$ [29]. All comparison algorithm codes are available on the platform PlatEMO [59]. The algorithm configuration is shown in Table 2.

For fairness, the population size and the number of function evaluations are equal for all algorithms. The population size is 100, and the maximum number of function evaluations is 10,000. Each algorithm uses single-point crossover and bitwise mutation to optimize binary vectors and simulated binary crossover and polynomial variation to optimize real vectors, so the crossover probability CR is set to 1, and the variation probability mu is set to $1/D$ [14].

After presenting the uniform parameter configuration for all algorithms, we will give the private parameter settings for each algorithm. In the method MOEADVA, the number of sampling solutions NCA in control variable analysis is set to 20. The number of solutions chosen for decision variable clustering in LMEA is five, the number of perturbations per solution is 50, and the number of solutions chosen for variable interaction analysis is five. The probability of the algorithm MOEAPSL generating offspring in the optimal subspace is 0.5, and if it is greater than 0.5, it generates offspring in the whole search space. SparseEA2 uses ordered grouping, and the number of groups is set to 4. Following the parameter settings of the classical reinforcement learning algorithm, we set the rewards in the proposed algorithm to 1 and -1 [49].

To better evaluate the algorithm performance, two evaluation metrics were used to measure the algorithm, HV [43] and IGD metrics [60, 61]. On each test instance, 30 times were run independently to obtain the mean and standard deviation. Statistical analysis was performed using the Wilcoxon rank sum test with a significance level of 0.05. The symbols " +", "-" and "=" indicate that our algorithm performed statistically better, worse, or similar to the comparison algorithms.

4.2 Parameter analysis and settings

The three parameters β, α, and ρ included in the proposed MOEADRL are discussed in this subsection, and the parameter settings for MOEADRL are then presented. The parameter β discusses which decision variables will be added to G_{zero}. We set β to 0.1, 0.2, 0.5, 0.8, and the more extreme 1.0. The HV and IGD values obtained by MOEADRL with different values of β are given in Tables 3 and 4. The objective is two, and the decision variable is 800.

Observing the values in Tables 3 and 4, it can be seen that the algorithm achieves the best results when β takes 0.8. This is because β takes 0.8 is more closely related to the problem dimension. The sparsity of our sparse problem is 0.1, which means that 90% of the decision variables is 0. The value of 0.8 is more closely related to the problem than 0.1 and 0.2, so it achieves a better result. When we adjust the parameter to the other extreme and add all variables to G_{zero}, it is clear that we will not get good results. This is because further variable mining is no longer possible after all variables are added to the group G_{zero}. The mining failure will affect the optimization of sparse variables. This can also support the validity of our three-way decision grouping strategy, which yields superior outcomes by delaying the unknown variables. So in the experimental part, we set the value of β to 0.8, which matches the properties of the problem and leaves some space for manipulation of the group G_{mix}.

The parameter α determines the composition of G_{one}. Since the problem sparsity is 0.1, it is a violation of the problem property for α to be greater than 0.1. We set α to 0.2 and 0.3 to verify this. In addition, we also set α to values of 0.03, 0.05, and 0.08 for comparison. The HV and IGD values obtained by the algorithm for different values of α are given in Tables 5 and 6. The objective is two, and the decision variable is 800.

In the eight test instances of Tables 5 and 6, α takes 0.1 can obtain the optimal value for all test items. In addition, comparing the test results of α greater than 0.1 and α less than 0.1, we can see that the results obtained by α take a value greater than 0.1 are inferior to those α takes a value less than 0.1. This fully indicates that the value that meets the sparsity of the problem can obtain better results, so the value of α for the experimental test part of this paper is 0.1.

The parameter ρ controls whether the offspring mask is generated using the three-way decision procedure or directly inherited from the parent. It is a balancing factor for constructing the offspring mask. In order to conduct the parameter sensitivity experiment, we set ρ to 0.1, 0.3, 0.5,

Table 3 HV values obtained by our algorithm with different values of β on SMOP1-SMOP8 in 30 runs

Problem	M	D	β=0.1	β=0.2	β=0.4	β=0.8	β=1
SMOP1	2	800	5.7125e-1 (5.26e-3)	5.6655e-1 (4.91e-3)	5.6704e-1 (3.46e-3)	5.6655e-1 (4.05e-3)	5.7016e-1 (1.37e-3)
SMOP2	2	800	5.4889e-1 (6.88e-3)	5.4803e-1 (5.59e-3)	5.5106e-1 (1.08e-2)	5.5587e-1 (8.79e-3)	5.5287e-1 (2.16e-2)
SMOP3	2	800	5.8157e-1 (6.21e-4)	5.8057e-1 (3.08e-3)	5.8000e-1 (3.12e-3)	5.8176e-1 (1.25e-4)	5.7974e-1 (2.99e-3)
SMOP4	2	800	8.1870e-1 (6.08e-5)	8.1872e-1 (3.81e-5)	8.1871e-1 (6.78e-5)	8.1875e-1 (5.66e-6)	8.1872e-1 (5.26e-5)
SMOP5	2	800	8.1847e-1 (4.62e-5)	8.1847e-1 (3.65e-5)	8.1825e-1 (8.09e-5)	8.1823e-1 (4.88e-5)	8.1818e-1 (1.52e-4)
SMOP6	2	800	8.1835e-1 (8.21e-5)	8.1836e-1 (6.98e-5)	8.1819e-1 (1.62e-4)	8.1747e-1 (1.44e-4)	8.1773e-1 (1.09e-4)
SMOP7	2	800	2.9878e-1 (1.30e-2)	2.9802e-1 (1.32e-2)	3.0982e-1 (1.83e-2)	3.1231e-1 (7.24e-3)	3.0436e-1 (6.99e-3)
SMOP8	2	800	1.7730e-1 (2.05e-2)	1.3796e-1 (1.93e-2)	1.8766e-1 (1.59e-2)	1.9587e-1 (3.59e-2)	1.6102e-1 (1.89e-2)

Table 4 IGD values obtained by our algorithm with different values of β on SMOP1-SMOP8 in 30 runs

Problem	M	D	β=0.1	β=0.2	β=0.4	β=0.8	β=1
SMOP1	2	800	1.0490e-2 (3.92e-3)	1.4045e-2 (3.88e-3)	1.3656e-2 (2.73e-3)	1.4047e-2 (3.16e-3)	1.1204e-2 (1.06e-3)
SMOP2	2	800	2.8147e-2 (5.46e-3)	2.8803e-2 (4.48e-3)	2.6358e-2 (8.63e-3)	2.2539e-2 (7.03e-3)	2.4961e-2 (1.72e-2)
SMOP3	2	800	3.8004e-3 (1.31e-4)	4.5102e-3 (1.76e-3)	4.7370e-3 (1.62e-3)	3.8208e-3 (5.35e-5)	4.8270e-3 (1.59e-3)
SMOP4	2	800	4.0743e-3 (8.16e-5)	4.1147e-3 (4.98e-5)	4.0811e-3 (1.04e-4)	4.0592e-3 (1.57e-5)	4.1017e-3 (7.52e-5)
SMOP5	2	800	4.1037e-3 (4.74e-5)	4.1298e-3 (5.13e-5)	4.1179e-3 (6.03e-5)	4.0946e-3 (7.89e-5)	4.1696e-3 (5.34e-5)
SMOP6	2	800	4.1686e-3 (8.60e-5)	4.1187e-3 (9.22e-5)	4.1450e-3 (8.90e-5)	4.3954e-3 (3.53e-5)	4.3089e-3 (3.83e-5)
SMOP7	2	800	3.7013e-2 (9.49e-3)	3.7524e-2 (9.66e-3)	2.8955e-2 (1.34e-2)	2.7070e-2 (5.31e-3)	3.2942e-2 (5.08e-3)
SMOP8	2	800	1.2957e-1 (1.92e-2)	1.7203e-1 (2.36e-2)	1.1947e-1 (1.39e-2)	1.1426e-1 (2.89e-2)	1.4578e-1 (1.94e-2)

Table 5 HV values obtained by our algorithm with different values of α on SMOP1-SMOP8 in 30 runs

Problem	α=0.03	α=0.05	α=0.08	α=0.1	α=0.2	α=0.3
SMOP1	5.5018e-1 (2.87e-3)	5.2481e-1 (3.19e-3)	5.1901e-1 (1.29e-3)	5.6655e-1 (4.05e-3)	5.1574e-1 (2.30e-3)	5.1604e-1 (1.84e-3)
SMOP2	4.8580e-1 (6.17e-3)	4.4892e-1 (3.09e-3)	4.3513e-1 (2.56e-3)	5.5587e-1 (8.79e-3)	4.3372e-1 (9.00e-4)	4.2908e-1 (1.67e-3)
SMOP3	5.5146e-1 (4.09e-3)	5.4570e-1 (4.56e-3)	5.2035e-1 (5.17e-3)	5.8176e-1 (1.25e-4)	5.1875e-1 (1.17e-3)	5.1489e-1 (3.68e-3)
SMOP4	8.1869e-1 (5.71e-5)	8.1864e-1 (1.44e-5)	8.1860e-1 (7.55e-5)	8.1875e-1 (5.66e-6)	8.1865e-1 (5.08e-5)	8.1869e-1 (4.08e-5)
SMOP5	8.1149e-1 (2.34e-4)	8.0764e-1 (3.28e-4)	7.8323e-1 (7.88e-4)	8.1823e-1 (4.88e-5)	7.8283e-1 (1.76e-3)	7.8396e-1 (8.86e-4)
SMOP6	8.0916e-1 (9.75e-4)	8.0354e-1 (6.01e-4)	7.8537e-1 (8.59e-4)	8.1747e-1 (1.44e-4)	7.8432e-1 (5.65e-4)	7.8448e-1 (1.13e-3)
SMOP7	1.9038e-1 (1.79e-2)	1.5729e-1 (6.06e-3)	1.4700e-1 (1.94e-3)	3.1231e-1 (7.24e-3)	1.3935e-1 (1.45e-3)	1.4105e-1 (1.15e-3)
SMOP8	4.7612e-2 (1.71e-2)	3.3463e-2 (8.53e-3)	3.0002e-2 (2.69e-3)	1.9587e-1 (3.59e-2)	2.9229e-2 (1.88e-3)	2.7317e-2 (5.15e-4)

Table 6 IGD values obtained by our algorithm with different values of α on SMOP1-SMOP8 in 30 runs

Problem	α=0.03	α=0.05	α=0.08	α=0.1	α=0.2	α=0.3
SMOP1	2.7060e-2 (2.28e-3)	4.6783e-2 (2.50e-3)	5.1321e-2 (9.23e-4)	1.4047e-2 (3.16e-3)	5.3899e-2 (1.92e-3)	5.3816e-2 (1.56e-3)
SMOP2	7.6526e-2 (4.84e-3)	1.0523e-1 (2.59e-3)	1.1683e-1 (2.10e-3)	2.2539e-2 (7.03e-3)	1.1789e-1 (7.24e-4)	1.2185e-1 (1.19e-3)
SMOP3	2.6079e-2 (3.31e-3)	3.0499e-2 (3.59e-3)	5.0423e-2 (4.13e-3)	3.8208e-3 (5.35e-5)	5.1642e-2 (9.48e-4)	5.4602e-2 (2.93e-3)
SMOP4	4.1162e-3 (9.72e-5)	4.1897e-3 (1.71e-5)	4.1758e-3 (9.22e-5)	4.0592e-3 (1.57e-5)	4.1910e-3 (6.25e-5)	4.1345e-3 (4.96e-5)
SMOP5	8.3512e-3 (1.08e-4)	1.1393e-2 (3.12e-4)	3.3132e-2 (9.30e-4)	4.0946e-3 (7.89e-5)	3.3615e-2 (1.64e-3)	3.2582e-2 (8.17e-4)
SMOP6	1.0364e-2 (8.59e-4)	1.5131e-2 (5.21e-4)	3.1417e-2 (8.71e-4)	4.3954e-3 (3.53e-5)	3.2443e-2 (4.21e-4)	3.2172e-2 (1.04e-3)
SMOP7	1.1624e-1 (1.49e-2)	1.4806e-1 (6.26e-3)	1.5913e-1 (2.46e-3)	2.7070e-2 (5.31e-3)	1.6755e-1 (1.66e-3)	1.6565e-1 (1.47e-3)
SMOP8	3.1820e-1 (3.91e-2)	3.5110e-1 (2.19e-2)	3.6065e-1 (7.95e-3)	1.1426e-1 (2.89e-2)	3.6246e-1 (6.42e-3)	3.6821e-1 (1.78e-3)

Table 7 HV values obtained by our algorithm with different values of ρ on SMOP1-SMOP8 in 30 runs

Problem	M	D	ρ=0.1	ρ=0.3	ρ=0.5	ρ=0.7	ρ=0.9
SMOP1	2	800	4.9927e-1 (1.58e-3)	5.0775e-1 (2.02e-3)	5.1291e-1 (4.01e-3)	5.6655e-1 (4.05e-3)	5.1920e-1 (1.78e-3)
SMOP2	2	800	4.0468e-1 (5.95e-3)	4.1640e-1 (1.68e-3)	4.2563e-1 (7.43e-4)	5.5587e-1 (8.79e-3)	4.2845e-1 (2.58e-3)
SMOP3	2	800	3.6538e-1 (2.24e-1)	3.5841e-1 (2.58e-1)	3.4070e-1 (2.95e-1)	5.8176e-1 (1.25e-4)	5.1801e-1 (9.78e-4)
SMOP4	2	800	8.1875e-1 (8.53e-5)	8.1875e-1 (4.95e-5)	8.1877e-1 (4.05e-5)	8.1875e-1 (5.66e-6)	8.1870e-1 (3.81e-5)
SMOP5	2	800	7.8098e-1 (2.94e-4)	7.8181e-1 (1.88e-3)	7.8298e-1 (2.09e-3)	8.1823e-1 (4.88e-5)	7.8351e-1 (2.25e-3)
SMOP6	2	800	7.8069e-1 (1.64e-3)	7.8285e-1 (8.04e-4)	7.8408e-1 (1.81e-3)	8.1747e-1 (1.44e-4)	7.8491e-1 (1.65e-3)
SMOP7	2	800	1.1545e-1 (1.69e-3)	1.3027e-1 (3.78e-3)	1.3428e-1 (2.87e-3)	3.1231e-1 (7.24e-3)	1.4017e-1 (3.46e-3)
SMOP8	2	800	1.5411e-2 (1.75e-3)	2.6259e-2 (2.76e-3)	2.8348e-2 (1.44e-3)	1.9587e-1 (3.59e-2)	2.9486e-2 (1.09e-3)

0.7, and 0.9. Tables 7 and 8 provide the HV and IGD values that the algorithm obtained from the different values of ρ. The objective is two, and the decision variable is 800.

When the value of ρ is taken from 0.1 to 0.7, it can be observed from Tables 7 and 8 that the algorithm can obtain better IGD and HV values as ρ increases. The IGD and HV values of the algorithm become worse when ρ is increased from 0.7 to 0.9. This is because when ρ is too small, the offspring frequently inherit the parent's mask before performing crossover mutation, and the algorithm concentrates on using the existing optimization results. When ρ is too large, the offspring directly adopt the current exploration result of the mask, and the algorithm focuses on exploring the value of the mask. When ρ is set to 0.7, the algorithm can achieve the balance between exploration and utilization. Therefore, the value of ρ sets 0.7 in our experiments.

4.3 Experimental results and analysis on benchmark problems

Table 9 shows the HV values of six algorithms on SMOP1-SMOP8 with two objectives and 300, 500, 800, and 1000 decision variables. In 32 test instances, our algorithms obtained 20 optimal values. In terms of statistical results, our algorithm significantly outperforms algorithms LMEA, MOEADVA, MOEAPSL, SparseEA, and SparseEA2 on 31, 32, 16, 21, and 17 test instances, respectively. Algorithms LMEA and MOEADVA did not obtain optimal values on any instance since these two algorithms are not customized for SMOPs. SparseEA also does not obtain optimal values, and although it is proposed for SMOPs, it is inferior to our algorithm. SparseEA2 obtained nine optimal values since it strengthened the connection between real and binary vectors. After flipping the binary vectors, the corresponding real vectors are optimized, resulting in a more efficient generation of sparse Pareto-optimal solutions than the algorithm SparseEA [29].

Table 10 shows the HV values of six algorithms on SMOP1-SMOP8 with three objectives and 300, 500, 800, and 1000 decision variables. In 32 test instances, our algorithms obtained 28 optimal values. Also, in terms of statistical results, our algorithm significantly outperforms algorithms LMEA, MOEADVA, MOEAPSL, SparseEA, and SparseEA2 on 31, 32, 22, 29, and 28 test instances, respectively. Algorithms LMEA, MOEADVA, and SparseEA also did not obtain optimal values on the test instances with three objectives. MOEAPSL and SparseEA2 obtained two optimal values, respectively. The algorithm MOEAPSL uses an unsupervised neural network to learn

Table 8 IGD values obtained by our algorithm with different values of ρ on SMOP1-SMOP8 in 30 runs

Problem	M	D	ρ=0.1	ρ=0.3	ρ=0.5	ρ=0.7	ρ=0.9
SMOP1	2	800	6.6810e-2 (1.09e-3)	6.0240e-2 (1.59e-3)	5.6114e-2 (3.17e-3)	1.4047e-2 (3.16e-3)	5.1171e-2 (1.15e-3)
SMOP2	2	800	1.4304e-1 (5.51e-3)	1.3287e-1 (1.42e-3)	1.2489e-1 (5.03e-4)	2.2539e-2 (7.03e-3)	1.2250e-1 (2.07e-3)
SMOP3	2	800	1.8939e-1 (2.06e-1)	2.0319e-1 (2.47e-1)	2.5620e-1 (3.44e-1)	3.8208e-3 (5.35e-5)	5.2127e-2 (6.32e-4)
SMOP4	2	800	4.0580e-3 (1.27e-4)	4.0339e-3 (8.47e-5)	4.0124e-3 (4.63e-5)	4.0592e-3 (1.57e-5)	4.0964e-3 (6.15e-5)
SMOP5	2	800	3.5445e-2 (3.38e-4)	3.4648e-2 (2.03e-3)	3.3560e-2 (2.17e-3)	4.0946e-3 (7.89e-5)	3.3191e-2 (2.03e-3)
SMOP6	2	800	3.5588e-2 (1.68e-3)	3.3799e-2 (9.08e-4)	3.2425e-2 (1.63e-3)	4.3954e-3 (3.53e-5)	3.1913e-2 (1.48e-3)
SMOP7	2	800	1.9824e-1 (2.63e-3)	1.7892e-1 (4.68e-3)	1.7368e-1 (3.20e-3)	2.7070e-2 (5.31e-3)	1.6637e-1 (4.47e-3)
SMOP8	2	800	4.1333e-1 (6.61e-3)	3.7271e-1 (8.93e-3)	3.6575e-1 (4.52e-3)	1.1426e-1 (2.89e-2)	3.6135e-1 (3.85e-3)

Table 9 HV values obtained by the six algorithms on SMOP1-SMOP8 with two objectives, the best results in each row are highlighted

Problem	D	LMEA	MOEADVA	MOEAPSL	SparseEA	SparseEA2	MOEADRL
SMOP1		5.7471e-1 (5.69e-4) =	4.9563e-1 (8.16e-4) -	5.8042e-1 (3.55e-4) =	5.7809e-1 (1.57e-3) =	5.7810e-1 (1.54e-3) =	5.7715e-1 (3.54e-3)
SMOP2		4.9602e-1 (5.61e-3) -	7.9993e-2 (3.52e-3) -	5.7814e-1 (3.43e-3) =	5.7364e-1 (8.46e-3) =	5.8117e-1 (3.15e-4)=	5.7333e-1 (8.66e-3)
SMOP3	300	2.8553e-2 (1.32e-3) -	7.2984e-3 (5.90e-4) -	3.5160e-1 (3.14e-1) =	5.7707e-1 (1.30e-3) -	5.7972e-1 (3.52e-4) =	5.8165e-1 (1.26e-3)
SMOP4		8.0133e-1 (1.90e-3) -	1.5415e-2 (6.07e-3) -	8.1797e-1 (2.13e-4) -	8.1807e-1 (2.07e-4) -	8.1795e-1 (2.38e-4) -	8.1873e-1 (2.86e-5)
SMOP5		4.2403e-1 (3.13e-3) -	4.2486e-1 (3.48e-5) -	8.1661e-1 (2.30e-4) -	8.1721e-1 (1.84e-4) -	8.1613e-1 (3.72e-4) -	8.1810e-1 (1.44e-4)
SMOP6		8.1004e-1 (6.40e-4) -	7.9993e-1 (1.29e-4) -	8.1593e-1 (4.99e-4) -	8.1689e-1 (1.81e-4) -	8.1571e-1 (6.16e-4) -	8.1805e-1 (5.94e-5)
SMOP7		1.5506e-1 (8.73e-2) -	4.1846e-2 (1.83e-3) -	3.4660e-1 (6.66e-5) =	3.2929e-1 (5.95e-3) =	3.4680e-1 (4.57e-5) =	3.3147e-1 (1.35e-2)
SMOP8		0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	1.7273e-1 (1.37e-2) =	1.5785e-1 (1.55e-2) -	1.6147e-1 (1.41e-2) =	2.0042e-1 (2.97e-2)
SMOP1		0.0000e+0 (0.00e+0) -	4.8592e-1 (4.21e-4) -	5.7973e-1 (1.49e-3) +	5.7376e-1 (4.41e-3) =	5.7741e-1 (5.24e-4) =	5.7585e-1 (3.56e-3)
SMOP2		0.0000e+0 (0.00e+0) -	1.0476e-1 (3.25e-3) -	5.7826e-1 (3.26e-3) +	5.5480e-1 (1.52e-2) =	5.8039e-1 (5.01e-4) +	5.6538e-1 (9.28e-3)
SMOP3	500	0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	5.8063e-1 (1.11e-3) =	5.7414e-1 (2.86e-3) -	5.7808e-1 (4.09e-4) -	5.8192e-1 (8.13e-4)
SMOP4		6.7912e-5 (1.52e-4) -	6.8513e-3 (9.66e-4) -	8.1786e-1 (2.53e-4) -	8.1815e-1 (3.75e-4) -	8.1811e-1 (2.21e-4) -	8.1874e-1 (3.99e-5)
SMOP5		3.5754e-4 (5.72e-4) -	4.2421e-1 (3.92e-5) -	8.1659e-1 (3.63e-4) -	8.1698e-1 (3.78e-4) -	8.1630e-1 (3.11e-4) -	8.1820e-1 (1.56e-4)
SMOP6		2.7066e-1 (4.00e-3) -	7.9561e-1 (1.35e-4) -	8.1608e-1 (2.84e-4) -	8.1655e-1 (3.75e-4) -	8.1506e-1 (2.65e-4) -	8.1797e-1 (1.70e-4)
SMOP7		0.0000e+0 (0.00e+0) -	6.2319e-2 (1.73e-3) -	3.4545e-1 (2.15e-3) +	3.1146e-1 (9.86e-3) =	3.4663e-1 (2.03e-4) +	3.1337e-1 (3.57e-3)
SMOP8		0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	1.6420e-1 (1.71e-2) =	1.6227e-1 (3.09e-2) =	1.5667e-1 (1.80e-2) =	1.8865e-1 (2.08e-2)
SMOP1		0.0000e+0 (0.00e+0) -	5.3299e-1 (2.31e-4) -	5.7994e-1 (9.46e-4) +	5.6743e-1 (2.65e-3) =	5.7697e-1 (4.10e-4) +	5.6655e-1 (4.05e-3)
SMOP2		0.0000e+0 (0.00e+0) -	2.4933e-1 (2.76e-3) -	5.7561e-1 (7.76e-4) +	5.3294e-1 (9.91e-4) -	5.7620e-1 (1.20e-3) +	5.5587e-1 (8.79e-3)
SMOP3	800	0.0000e+0 (0.00e+0) -	2.0526e-3 (3.04e-4) -	5.7868e-1 (4.45e-3) -	5.6887e-1 (3.72e-3) -	5.7780e-1 (4.22e-4) -	5.8176e-1 (1.25e-4)
SMOP4		1.5982e-1 (3.57e-1) -	4.7820e-3 (1.45e-3) -	8.1743e-1 (6.68e-4) -	8.1808e-1 (2.48e-4) -	8.1794e-1 (1.97e-4) -	8.1875e-1 (5.66e-6)
SMOP5		0.0000e+0 (0.00e+0) -	4.2612e-1 (1.16e-5) -	8.1638e-1 (1.71e-4) -	8.1702e-1 (1.64e-4) -	8.1633e-1 (2.26e-4) -	8.1823e-1 (4.88e-5)
SMOP6		2.6788e-1 (4.14e-3) -	8.0403e-1 (4.23e-5) -	8.1609e-1 (3.74e-4) -	8.1631e-1 (3.45e-4) -	8.1467e-1 (7.00e-4) -	8.1747e-1 (1.44e-4)
SMOP7		0.0000e+0 (0.00e+0) -	9.6809e-2 (8.31e-4) -	3.3737e-1 (1.79e-2) +	2.9016e-1 (5.93e-3) =	3.4577e-1 (1.77e-3) +	3.1231e-1 (7.24e-3)
SMOP8		0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	1.4907e-1 (1.55e-2) -	1.3451e-1 (9.71e-3) -	1.3736e-1 (1.31e-2) -	1.9587e-1 (3.59e-2)
SMOP1		0.0000e+0 (0.00e+0) -	5.4820e-1 (1.48e-4) -	5.7667e-1 (1.03e-3) +	5.6508e-1 (1.45e-3) =	5.7696e-1 (4.97e-4) +	5.6900e-1 (3.03e-3)
SMOP2		0.0000e+0 (0.00e+0) -	3.2105e-1 (1.25e-3) -	5.5847e-1 (3.35e-3) +	5.3195e-1 (5.12e-3) =	5.7495e-1 (3.91e-4) +	5.3996e-1 (8.33e-3)
SMOP3	1000	0.0000e+0 (0.00e+0) -	5.3247e-3 (2.87e-4) -	5.7964e-1 (6.66e-4) -	5.6651e-1 (2.99e-3) -	5.7812e-1 (4.82e-4) -	5.8163e-1 (2.62e-4)
SMOP4		0.0000e+0 (0.00e+0) -	2.7485e-3 (8.44e-4) -	8.1742e-1 (2.20e-4) -	8.0218e-1 (2.21e-2) -	8.1810e-1 (1.24e-4) -	8.1872e-1 (5.57e-5)
SMOP5		0.0000e+0 (0.00e+0) -	4.2657e-1 (1.99e-5) -	8.1347e-1 (4.77e-4) -	8.1696e-1 (1.31e-4) -	8.1624e-1 (2.80e-4) -	8.1817e-1 (9.48e-5)
SMOP6		2.6686e-1 (7.03e-3) -	8.0686e-1 (5.30e-5) -	8.1097e-1 (5.85e-4) -	8.1626e-1 (2.05e-4) -	8.1459e-1 (1.50e-4) -	8.1732e-1 (2.84e-4)
SMOP7		0.0000e+0 (0.00e+0) -	1.1144e-1 (1.30e-3) -	2.3915e-1 (8.77e-2) =	2.7321e-1 (5.81e-3) -	3.4471e-1 (2.05e-3) +	2.8579e-1 (8.33e-3)
SMOP8		0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	1.2973e-1 (5.44e-3) -	1.2354e-1 (1.10e-2) -	1.3688e-1 (1.64e-3) -	1.7937e-1 (2.05e-2)
+/-/=		0/31/1	0/32/0	8/16/8	0/21/11	8/17/7	

the Pareto-optimal subspace, and its genetic operators are performed in the learned subspace rather than in the original search space. As a result, the search space is reduced, resulting in better results than other algorithms but much worse than our algorithm [14]. Comparing Tables 9 and 10, we can see that our algorithm performs significantly better than other MOEAs in dealing with large-scale SMOPs, and the higher the number of objectives, the better the performance of our algorithm. Our algorithm greatly enhances the convergence of the population toward the Pareto-optimal solution, not only considering the mining of sparse variables through reinforcement learning but also designing a three-way grouping method of decision variables for optimization. Experimental results also demonstrate the superiority of our algorithm in sparse variable mining and *dec* optimization.

Tables 11 and 12 show the IGD values of the six algorithms on SMOP1-SMOP8 with two and three objectives, respectively. In Table 11, our algorithms obtain the optimal values on 20 test instances. In terms of statistical results, the proposed algorithm outperforms algorithms LMEA, MOEADVA, MOEAPSL, SparseEA, and SparseEA2 on 31, 32, 18, 21, and 18 instances, respectively. In Table 12, our algorithm obtains the optimal value on 25 test instances. In terms of statistical results, the proposed algorithm outperforms algorithms LMEA, MOEADVA, MOEAPSL, SparseEA, and SparseEA2 on 30, 31, 22, 27, and 26 instances, respectively. This is similar to the results obtained by using the HV evaluation criteria. The test results on both IGD and HV demonstrate that our algorithm is more effective than the comparison algorithms in solving large-scale SMOPs. In addition, the experimental results show that most algorithms designed for the sparse problem work significantly better than traditional MOEAs. In order to further demonstrate the convergence performance and population distribution of our algorithm, we next compare the convergence curves and population distributions of the six algorithms.

Table 10 HV values obtained by the six algorithms on SMOP1-SMOP8 with three objectives, the best results in each row are highlighted

Problem	D	LMEA	MOEADVA	MOEAPSL	SparseEA	SparseEA2	MOEADRL
SMOP1		8.1764e-1 (6.62e-3) =	7.7026e-1 (3.48e-4) -	8.1677e-1 (3.17e-3) -	8.0718e-1 (3.54e-3) -	8.0333e-1 (6.79e-3) -	8.2540e-1 (2.09e-3)
SMOP2		7.7688e-1 (6.80e-3) -	3.1033e-1 (5.63e-3) -	8.0387e-1 (3.94e-3) -	7.7826e-1 (1.25e-2) -	7.8542e-1 (6.27e-3) -	8.2232e-1 (3.80e-3)
SMOP3	300	2.2187e-1 (4.56e-3) -	1.4595e-1 (8.23e-4) -	8.1939e-1 (2.82e-3) -	8.0934e-1 (2.86e-3) -	8.0845e-1 (2.18e-3) -	8.3948e-1 (3.01e-4)
SMOP4		9.6926e-1 (9.47e-4) -	3.6469e-1 (1.77e-2) -	9.3322e-1 (1.33e-2) -	9.6479e-1 (2.02e-3) -	9.6716e-1 (6.64e-4) -	9.7578e-1 (8.77e-5)
SMOP5		7.3088e-1 (2.45e-1) -	8.3749e-1 (3.03e-5) -	9.2708e-1 (4.36e-3) -	9.6253e-1 (2.97e-3) -	9.6574e-1 (8.32e-4) -	9.7523e-1 (1.31e-4)
SMOP6		9.7233e-1 (9.36e-4) -	9.7062e-1 (3.25e-5) -	9.3614e-1 (1.62e-2) -	9.6230e-1 (4.37e-3) -	9.6457e-1 (1.90e-3) -	9.7490e-1 (1.85e-4)
SMOP7		3.6532e-1 (5.79e-3) -	1.2170e-1 (4.60e-3) -	5.1369e-1 (8.72e-3) =	4.9145e-1 (8.42e-3) -	5.0023e-1 (8.04e-3) =	5.1484e-1 (1.49e-2)
SMOP8		0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	3.1092e-1 (3.63e-2) -	3.0201e-1 (3.03e-2) -	2.8050e-1 (1.69e-2) -	3.8591e-1 (1.98e-2)
SMOP1		0.0000e+0 (0.00e+0) -	7.6471e-1 (4.36e-4) -	8.1563e-1 (3.41e-3) -	8.0088e-1 (3.79e-3) -	8.0555e-1 (4.84e-3) -	8.2329e-1 (2.10e-3)
SMOP2		0.0000e+0 (0.00e+0) -	3.5974e-1 (5.31e-3) -	7.9379e-1 (3.21e-3) -	7.7279e-1 (1.08e-2) -	7.8290e-1 (3.94e-3) -	8.0598e-1 (7.13e-3)
SMOP3	500	0.0000e+0 (0.00e+0) -	7.9947e-2 (9.29e-4) -	6.5587e-1 (3.65e-1) -	7.9580e-1 (5.11e-3) -	8.0483e-1 (4.43e-3) -	8.3859e-1 (6.28e-4)
SMOP4		2.6830e-1 (2.80e-2) -	3.2167e-1 (2.45e-2) -	9.2879e-1 (9.70e-3) -	9.6213e-1 (3.43e-3) -	9.6799e-1 (2.60e-3) -	9.7577e-1 (7.15e-5)
SMOP5		2.7333e-1 (3.44e-2) -	8.3709e-1 (2.23e-5) -	9.2577e-1 (4.48e-3) -	9.6515e-1 (1.57e-3) -	9.6537e-1 (1.09e-3) -	9.7546e-1 (1.15e-4)
SMOP6		7.1958e-1 (5.77e-3) -	9.6970e-1 (4.74e-5) -	9.2341e-1 (3.22e-3) -	9.6079e-1 (1.87e-3) -	9.6557e-1 (9.58e-4) -	9.7509e-1 (1.18e-4)
SMOP7		0.0000e+0 (0.00e+0) -	1.6835e-1 (1.09e-3) -	5.0397e-1 (7.16e-3) +	4.4811e-1 (3.69e-3) =	4.8768e-1 (3.32e-3) =	4.6534e-1 (2.15e-2)
SMOP8		0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	2.8000e-1 (3.29e-2) =	2.5172e-1 (6.14e-3) -	2.4204e-1 (2.82e-2) -	2.9808e-1 (2.21e-2)
SMOP1		0.0000e+0 (0.00e+0) -	7.9541e-1 (3.30e-4) -	5.3573e-1 (3.76e-1) -	8.0325e-1 (3.73e-3) -	8.0962e-1 (4.55e-3) -	8.1825e-1 (2.17e-3)
SMOP2		0.0000e+0 (0.00e+0) -	5.5734e-1 (2.41e-3) -	7.9028e-1 (9.03e-3) =	7.6169e-1 (5.67e-3) -	7.7648e-1 (9.34e-3) -	7.9719e-1 (5.49e-3)
SMOP3	800	0.0000e+0 (0.00e+0) -	1.2571e-1 (6.79e-4) -	8.1652e-1 (2.40e-3) -	7.9381e-1 (8.42e-3) -	8.0642e-1 (4.31e-3) -	8.3088e-1 (6.24e-3)
SMOP4		2.5703e-1 (3.74e-2) -	3.2080e-1 (1.76e-2) -	9.5939e-1 (1.05e-2) -	9.6486e-1 (1.92e-3) -	9.6815e-1 (1.78e-3) -	9.7589e-1 (9.10e-5)
SMOP5		2.6259e-1 (2.55e-2) -	8.3809e-1 (9.16e-6) -	9.2612e-1 (2.65e-3) -	9.6303e-1 (1.99e-3) -	9.6663e-1 (1.51e-3) -	9.7541e-1 (2.14e-5)
SMOP6		7.2306e-1 (1.09e-2) -	9.7153e-1 (3.61e-6) -	9.3745e-1 (2.00e-2) -	9.6232e-1 (4.00e-3) -	9.6413e-1 (1.13e-3) -	9.7489e-1 (1.73e-4)
SMOP7		0.0000e+0 (0.00e+0) -	2.2975e-1 (1.65e-3) -	4.6268e-1 (4.51e-2) =	4.1899e-1 (1.41e-2) =	4.8950e-1 (3.31e-3) +	4.4807e-1 (1.83e-2)
SMOP8		0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	2.0630e-1 (4.87e-2) =	2.1170e-1 (2.28e-2) =	1.9490e-1 (2.91e-2) -	2.3871e-1 (1.45e-2)
SMOP1		0.0000e+0 (0.00e+0) -	8.0503e-1 (9.85e-5) -	8.2027e-1 (4.42e-3) =	7.9695e-1 (5.94e-3) -	8.0668e-1 (3.07e-3) -	8.1392e-1 (2.96e-3)
SMOP2		0.0000e+0 (0.00e+0) -	6.3352e-1 (1.35e-3) -	7.9451e-1 (6.56e-3) =	7.5411e-1 (1.08e-2) -	7.6257e-1 (4.54e-3) -	7.9516e-1 (4.31e-3)
SMOP3	1000	0.0000e+0 (0.00e+0) -	1.4451e-1 (3.53e-4) -	8.1561e-1 (5.26e-3) -	7.8478e-1 (3.53e-3) -	8.0234e-1 (6.10e-3) -	8.2581e-1 (5.79e-3)
SMOP4		2.4248e-1 (2.78e-2) -	3.2843e-1 (9.97e-3) -	9.4123e-1 (1.91e-2) -	9.6391e-1 (2.80e-3) -	9.6694e-1 (2.97e-3) -	9.7589e-1 (9.47e-5)
SMOP5		2.7076e-1 (2.23e-2) -	8.3832e-1 (3.48e-6) -	9.2800e-1 (1.01e-2) -	9.6369e-1 (2.00e-3) -	9.6681e-1 (1.07e-3) -	9.7541e-1 (1.18e-4)
SMOP6		7.1505e-1 (1.20e-2) -	9.7216e-1 (6.12e-6) -	9.3360e-1 (1.73e-2) -	9.6343e-1 (1.90e-3) -	9.6298e-1 (2.37e-3) -	9.7499e-1 (1.67e-4)
SMOP7		0.0000e+0 (0.00e+0) -	2.5609e-1 (1.28e-3) -	4.7297e-1 (1.32e-2) +	4.0673e-1 (1.77e-2) -	4.7632e-1 (3.62e-3) +	4.4327e-1 (3.40e-3)
SMOP8		0.0000e+0 (0.00e+0) -	0.0000e+0 (0.00e+0) -	2.3643e-1 (3.46e-2) =	2.0326e-1 (1.66e-2) -	1.6704e-1 (1.45e-2) -	2.4831e-1 (3.13e-2)
+/-/=		0/31/1	0/32/0	2/22/8	0/29/3	2/28/2	

4.4 Algorithm convergence and population distributivity analysis

The convergence curves of the IGD values of six algorithms are shown in Figs. 4 and 5 to help clarify the experimental results. The decision variable is 300, and the objective is two. As can be seen from these figures, the algorithm MOEADVA only converges to the relatively worse IGD value on both SMOP4 and SMOP5. LMEA requires a vast number of evaluations on both SMOP4 and SMOP5 to converge to the poor IGD value. This is because neither of these algorithms is proposed for SMOPs. The algorithm MOEAPSL converges unstably on SMOP4, and the IGD value deteriorates during the convergence process. SparseEA and SparseEA2 have better convergence performance than other algorithms but are inferior to ours. Our algorithm obtains lower average IGD values on both SMOP4 and SMOP5, which indicates that our algorithm has optimal performance and efficiency. In addition, the proposed algorithm is also the fastest to converge to the optimal value, which indicates that our algorithm also outperforms other algorithms in terms of convergence speed.

Since the performance of LMEA and MOEADVA on sparse problems are far from ours, and both algorithms have difficulty converging on sparse problems with large-scale variables, we choose sparse problems with 300 decision variables to compare convergence and population distributivity.

The population distributivity is significant for the algorithm. We plot the Pareto-optimal fronts with median IGD values obtained by the six algorithms in Figs. 6 and 7. The details of the population distribution are enlarged in the figure. The number of decision variables is 300, and the objective is two. It can be observed that our algorithms obtain solutions with good convergence and distributivity. When approaching the true PF, we mine the sparse variables while optimizing the nonzero decision variables, greatly reducing the search space. Algorithms LMEA and

Table 11 IGD values obtained by the six algorithms on SMOP1-SMOP8 with two objectives, the best results in each row are highlighted

Problem	D	LMEA	MOEADVA	MOEAPSL	SparseEA	SparseEA2	MOEADRL
SMOP1		7.9517e-3 (4.80e-4) =	6.6401e-2 (3.48e-4) -	4.8182e-3 (2.09e-4) =	5.8711e-3 (8.04e-4) =	5.9299e-3 (8.57e-4) =	6.3883e-3 (2.05e-3)
SMOP2		6.4980e-2 (4.54e-3) -	5.0229e-1 (8.08e-3) -	6.0791e-3 (1.73e-3) =	9.1642e-3 (5.74e-3) =	4.5891e-3 (2.17e-4) =	9.5363e-3 (5.69e-3)
SMOP3	300	6.5297e-1 (5.33e-4) -	7.3337e-1 (2.01e-3) -	3.0089e-1 (4.06e-1) -	6.3920e-3 (7.94e-4) -	5.0890e-3 (1.83e-4) -	3.9819e-3 (4.22e-4)
SMOP4		1.6662e-2 (1.53e-3) -	9.3260e-1 (2.62e-2) -	4.9269e-3 (3.36e-4) -	4.8057e-3 (2.34e-4) -	4.9180e-3 (3.26e-4) -	4.0900e-3 (4.42e-5)
SMOP5		3.4723e-1 (2.37e-4) -	3.4914e-1 (8.63e-5) -	5.4127e-3 (3.27e-4) -	4.9289e-3 (1.42e-4) -	5.4035e-3 (2.82e-4) -	4.1989e-3 (9.64e-5)
SMOP6		1.0052e-2 (7.27e-4) -	1.8594e-2 (1.39e-4) -	5.8643e-3 (6.21e-4) -	5.0590e-3 (1.79e-4) -	5.6012e-3 (3.69e-4) -	4.2317e-3 (3.85e-5)
SMOP7		6.4471e-1 (1.19e+0) -	3.1445e-1 (3.16e-3) -	5.1014e-3 (6.47e-5) =	1.4859e-2 (4.18e-3) =	5.0173e-3 (2.45e-4) =	1.3766e-2 (9.00e-3)
SMOP8		8.8307e-1 (2.74e-2) -	1.6401e+0 (1.51e-2) -	1.3326e-1 (1.40e-2) =	1.4858e-1 (1.67e-2) -	1.4463e-1 (1.51e-2) =	1.1021e-1 (2.51e-2)
SMOP1		1.5492e+0 (2.68e-2) -	7.5436e-2 (5.25e-4) -	5.1166e-3 (6.50e-4) =	8.6340e-3 (3.02e-3) =	6.2034e-3 (3.15e-4) =	7.1315e-3 (2.39e-3)
SMOP2		2.1901e+0 (1.17e-2) -	4.6511e-1 (2.80e-3) -	5.9850e-3 (1.75e-3) +	2.2898e-2 (1.18e-2) =	4.8692e-3 (2.40e-4) +	1.5026e-2 (7.23e-3)
SMOP3	500	2.5672e+0 (2.27e-2) -	8.6484e-1 (2.89e-3) -	4.6863e-3 (3.57e-4) -	8.3129e-3 (1.84e-3) -	5.8088e-3 (2.23e-4) -	3.8402e-3 (2.27e-4)
SMOP4		1.0922e+0 (6.09e-3) -	9.8087e-1 (1.76e-2) -	5.0897e-3 (3.77e-4) -	4.5635e-3 (4.01e-4) -	4.7225e-3 (2.63e-4) -	4.0862e-3 (7.35e-5)
SMOP5		1.0586e+0 (1.22e-2) -	3.4975e-1 (4.92e-5) -	5.2576e-3 (2.90e-4) -	4.9229e-3 (3.29e-4) -	5.3229e-3 (3.60e-4) -	4.1255e-3 (8.39e-5)
SMOP6		4.8504e-1 (4.94e-3) -	2.2545e-2 (1.40e-4) -	5.5984e-3 (3.63e-4) -	5.0909e-3 (2.47e-4) -	5.9736e-3 (2.86e-4) -	4.2240e-3 (1.04e-4)
SMOP7		2.8364e+0 (2.82e-2) -	2.7524e-1 (2.05e-3) -	5.4541e-3 (7.94e-4) +	2.7418e-2 (6.98e-3) =	5.0375e-3 (1.71e-4) +	2.6310e-2 (2.63e-3)
SMOP8		3.6867e+0 (1.14e-2) -	1.5554e+0 (6.77e-3) -	1.4198e-1 (1.77e-2) =	1.4528e-1 (3.20e-2) =	1.5025e-1 (1.92e-2) =	1.1934e-1 (1.85e-2)
SMOP1		1.5933e+0 (2.34e-2) -	4.0253e-2 (2.68e-4) -	5.0079e-3 (5.69e-4) +	1.3058e-2 (2.00e-3) =	6.4311e-3 (2.62e-4) +	1.4047e-2 (3.16e-3)
SMOP2		2.2241e+0 (1.09e-2) -	2.8124e-1 (4.42e-3) -	7.2437e-3 (4.81e-4) +	4.0055e-2 (7.87e-4) -	6.9876e-3 (7.57e-4) +	2.2539e-2 (7.03e-3)
SMOP3	800	2.5886e+0 (9.15e-3) -	7.8745e-1 (1.48e-3) -	5.8788e-3 (2.63e-3) -	1.2020e-2 (2.75e-3) -	5.9254e-3 (2.69e-4) -	3.8208e-3 (5.35e-5)
SMOP4		8.8376e-1 (4.84e-1) -	1.0076e+0 (1.70e-2) -	5.6757e-3 (9.13e-4) -	4.7608e-3 (3.15e-4) -	4.9539e-3 (2.29e-4) -	4.0592e-3 (1.57e-5)
SMOP5		1.0781e+0 (1.57e-2) -	3.4801e-1 (2.01e-5) -	5.4667e-3 (2.85e-4) -	4.8792e-3 (4.33e-5) -	5.2982e-3 (2.60e-4) -	4.0946e-3 (7.89e-5)
SMOP6		4.9000e-1 (4.14e-3) -	1.4929e-2 (2.35e-5) -	5.6558e-3 (3.14e-4) -	5.2381e-3 (3.27e-4) -	6.1432e-3 (6.06e-4) -	4.3954e-3 (3.53e-5)
SMOP7		2.9140e+0 (2.76e-2) -	2.2132e-1 (7.84e-4) -	1.0676e-2 (1.19e-2) +	4.2710e-2 (4.17e-3) =	5.4037e-3 (7.10e-4) +	2.7070e-2 (5.31e-3)
SMOP8		3.6993e+0 (5.09e-3) -	1.4154e+0 (6.96e-3) -	1.5803e-1 (1.75e-2) -	1.7378e-1 (1.15e-2) -	1.7066e-1 (1.60e-2) -	1.1426e-1 (2.89e-2)
SMOP1		1.5942e+0 (1.84e-2) -	2.8596e-2 (1.05e-4) -	6.6987e-3 (6.27e-4) +	1.4921e-2 (1.16e-3) =	6.4427e-3 (2.99e-4) +	1.2122e-2 (2.38e-3)
SMOP2		2.2288e+0 (9.78e-3) -	2.1147e-1 (1.68e-3) -	1.9932e-2 (2.65e-3) +	4.0978e-2 (4.06e-3) =	7.7945e-3 (2.55e-4) +	3.5306e-2 (6.62e-3)
SMOP3	1000	2.5805e+0 (6.29e-3) -	7.5824e-1 (2.67e-4) -	4.9784e-3 (3.09e-4) -	1.3830e-2 (2.29e-3) -	5.7554e-3 (3.15e-4) -	3.8339e-3 (6.99e-5)
SMOP4		1.1021e+0 (3.01e-3) -	1.0163e+0 (7.31e-3) -	5.5031e-3 (3.67e-4) -	1.9908e-2 (2.07e-2) -	4.7214e-3 (1.77e-4) -	4.1094e-3 (8.47e-5)
SMOP5		1.0802e+0 (1.20e-2) -	3.4757e-1 (1.90e-5) -	7.0472e-3 (3.74e-4) -	4.9753e-3 (1.72e-4) -	5.3773e-3 (2.85e-4) -	4.1337e-3 (3.11e-5)
SMOP6		4.9360e-1 (6.16e-3) -	1.2410e-2 (6.58e-5) -	9.1245e-3 (6.41e-4) -	5.3698e-3 (2.53e-4) -	6.2800e-3 (1.71e-4) -	4.4595e-3 (1.34e-4)
SMOP7		2.9517e+0 (4.38e-2) -	2.0023e-1 (9.90e-4) -	8.1480e-2 (6.47e-2) =	5.4980e-2 (4.20e-3) -	5.9555e-3 (1.08e-3) +	4.6444e-2 (6.08e-3)
SMOP8		3.7074e+0 (7.40e-3) -	1.3321e+0 (1.69e-2) -	1.7995e-1 (6.83e-3) -	1.8729e-1 (1.38e-2) -	1.7113e-1 (2.64e-3) -	1.2772e-1 (1.90e-2)
+/-/=		0/31/1	0/32/0	7/18/7	0/21/11	8/18/6	

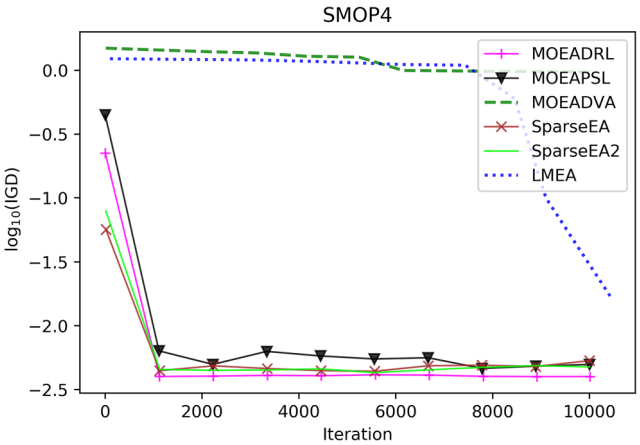


Fig. 4 Convergence comparison of six algorithms on SMOP4 with the objective of two and decision variable of 300

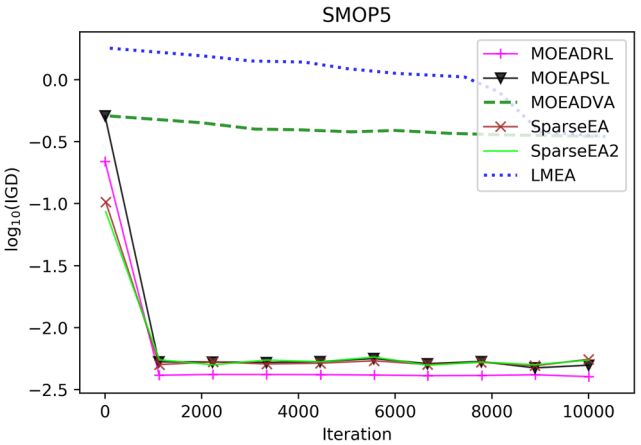


Fig. 5 Convergence comparison of six algorithms on SMOP5 with the objective of two and decision variable of 300

MOEADVA are far from the true PF; MOEADVA has the worst population distributivity among all algorithms, with the most uneven distributivity of individuals. Although the populations of algorithms MOEAPSL, SparseEA, and SparseEA2 converge closer to the true PF than the above two algorithms, they are also significantly inferior to our

Table 12 IGD values obtained by the six algorithms on SMOP1-SMOP8 with three objectives, the best results in each row are highlighted

Problem	D	LMEA	MOEADVA	MOEAPSL	SparseEA	SparseEA2	MOEADRL
SMOP1		4.5731e-2 (1.94e-3) +	7.8933e-2 (3.60e-4) -	5.6702e-2 (1.19e-3) -	6.1796e-2 (2.99e-3) -	6.2991e-2 (4.15e-3) -	4.7934e-2 (1.33e-3)
SMOP2		7.2962e-2 (4.65e-3) -	4.4402e-1 (6.68e-3) -	6.3421e-2 (4.50e-3) -	8.2060e-2 (8.21e-3) -	7.6099e-2 (3.25e-3) -	5.0183e-2 (2.36e-3)
SMOP3	300	5.5024e-1 (1.95e-3) -	6.2868e-1 (5.12e-4) -	5.8496e-2 (2.87e-3) -	6.0880e-2 (1.80e-3) -	6.2144e-2 (5.91e-4) -	4.0782e-2 (2.83e-4)
SMOP4		3.3758e-2 (8.61e-4) -	5.8565e-1 (5.32e-3) -	1.3841e-1 (2.87e-2) -	6.4128e-2 (5.91e-3) -	5.8709e-2 (3.22e-3) -	3.0610e-2 (7.76e-4)
SMOP5		3.0767e-1 (2.08e-1) -	2.1765e-1 (4.39e-5) -	1.5045e-1 (6.80e-3) -	7.1065e-2 (1.02e-2) -	5.8789e-2 (4.16e-3) -	3.0658e-2 (4.14e-4)
SMOP6		3.2214e-2 (3.04e-3) =	3.9309e-2 (6.64e-5) -	1.3060e-1 (3.63e-2) -	6.8348e-2 (1.07e-2) -	6.1400e-2 (5.79e-3) -	3.0824e-2 (7.97e-4)
SMOP7		1.4170e-1 (6.88e-3) -	3.3664e-1 (2.64e-3) -	7.4966e-2 (1.32e-3) -	8.4325e-2 (2.29e-3) -	8.5454e-2 (5.31e-3) -	6.7479e-2 (4.94e-3)
SMOP8		9.6142e-1 (2.46e-2) -	1.6629e+0 (2.76e-2) -	1.7402e-1 (2.48e-2) -	1.8308e-1 (2.07e-2) -	1.9487e-1 (1.51e-2) -	1.3602e-1 (1.19e-2)
SMOP1		1.3045e+0 (5.27e-2) -	8.3509e-2 (3.70e-4) -	5.8511e-2 (1.81e-3) -	6.7252e-2 (6.24e-3) -	6.0956e-2 (1.53e-3) -	4.9009e-2 (1.74e-3)
SMOP2		1.8407e+0 (1.78e-2) -	4.0859e-1 (4.38e-3) -	6.8467e-2 (2.08e-3) =	8.5975e-2 (6.63e-3) -	7.9456e-2 (3.68e-3) -	6.2517e-2 (5.96e-3)
SMOP3	500	2.1533e+0 (1.50e-2) -	7.2761e-1 (1.82e-3) -	2.6306e-1 (4.57e-1) -	6.9459e-2 (5.18e-3) -	6.4323e-2 (1.68e-3) -	4.2388e-2 (5.27e-4)
SMOP4		6.8750e-1 (1.56e-2) -	6.1511e-1 (2.37e-2) -	1.4626e-1 (1.83e-2) -	7.4027e-2 (1.27e-2) -	5.7931e-2 (7.18e-3) -	2.9994e-2 (4.97e-4)
SMOP5		6.8294e-1 (3.39e-2) -	2.1805e-1 (8.49e-6) -	1.5403e-1 (5.85e-3) -	6.1288e-2 (6.70e-3) -	6.0465e-2 (2.83e-3) -	3.0112e-2 (5.45e-4)
SMOP6		3.1098e-1 (7.27e-3) -	4.0476e-2 (7.51e-5) -	1.5710e-1 (3.68e-3) -	7.3675e-2 (4.41e-3) -	5.9046e-2 (3.30e-3) -	3.0377e-2 (3.88e-4)
SMOP7		2.9197e+0 (6.07e-2) -	2.9381e-1 (1.08e-3) -	7.7246e-2 (4.21e-3) =	1.0296e-1 (3.36e-3) =	9.2324e-2 (3.43e-3) =	9.0533e-2 (1.11e-2)
SMOP8		3.7102e+0 (1.09e-2) -	1.5818e+0 (1.73e-2) -	1.9693e-1 (2.38e-2) =	2.1466e-1 (4.56e-3) =	2.2406e-1 (1.82e-2) =	1.9513e-1 (1.69e-2)
SMOP1		1.3276e+0 (8.79e-3) -	5.9913e-2 (1.05e-4) -	2.7919e-1 (3.07e-1) -	6.4371e-2 (3.14e-3) -	5.8895e-2 (2.76e-3) -	5.2800e-2 (1.57e-3)
SMOP2		1.8468e+0 (1.02e-2) -	2.5285e-1 (2.05e-3) -	7.1061e-2 (3.32e-3) =	9.3234e-2 (5.41e-3) -	8.0112e-2 (4.68e-3) =	7.0028e-2 (4.92e-3)
SMOP3	800	2.1554e+0 (5.84e-3) -	6.5778e-1 (7.09e-4) -	5.9254e-2 (1.18e-3) -	7.4195e-2 (6.17e-3) -	6.2659e-2 (3.65e-3) -	4.5581e-2 (3.59e-3)
SMOP4		6.9762e-1 (3.76e-2) -	6.3249e-1 (1.58e-2) -	7.6714e-2 (2.60e-2) -	6.1667e-2 (5.22e-3) -	5.5860e-2 (5.83e-3) -	3.0004e-2 (4.84e-4)
SMOP5		6.7751e-1 (4.83e-3) -	2.1701e-1 (1.92e-5) -	1.5256e-1 (3.56e-3) -	6.6200e-2 (6.65e-3) -	6.0620e-2 (7.08e-3) -	3.0093e-2 (6.11e-4)
SMOP6		3.1725e-1 (1.54e-2) -	3.8218e-2 (7.13e-6) -	1.2456e-1 (4.47e-2) -	6.8220e-2 (1.04e-2) -	6.3196e-2 (6.24e-3) -	3.0796e-2 (5.21e-4)
SMOP7		2.9639e+0 (3.35e-2) -	2.3980e-1 (5.93e-4) -	9.5273e-2 (2.22e-2) =	1.1679e-1 (8.48e-3) =	9.1015e-2 (1.99e-3) =	1.0031e-1 (1.06e-2)
SMOP8		3.7260e+0 (2.65e-3) -	1.4403e+0 (7.29e-3) -	2.5735e-1 (4.17e-2) =	2.4796e-1 (1.63e-2) =	2.6551e-1 (2.39e-2) -	2.3865e-1 (1.09e-2)
SMOP1		1.3496e+0 (1.46e-2) -	5.3707e-2 (5.36e-5) =	5.5290e-2 (2.48e-3) =	6.9815e-2 (3.56e-3) -	5.9966e-2 (2.99e-3) =	5.6338e-2 (2.22e-3)
SMOP2		1.8479e+0 (1.32e-2) -	1.9347e-1 (1.54e-3) -	6.8511e-2 (4.15e-3) =	9.3063e-2 (5.06e-3) -	8.9664e-2 (2.72e-3) -	7.1807e-2 (3.20e-3)
SMOP3	1000	2.1497e+0 (2.06e-2) -	6.3274e-1 (3.31e-4) -	5.8960e-2 (2.29e-3) -	7.6265e-2 (4.56e-3) -	6.4014e-2 (4.28e-3) -	4.8252e-2 (3.02e-3)
SMOP4		6.9909e-1 (2.24e-2) -	6.2599e-1 (1.18e-2) -	1.1972e-1 (4.22e-2) -	6.9429e-2 (1.00e-2) -	5.9615e-2 (9.12e-3) -	2.9761e-2 (3.05e-4)
SMOP5		6.7564e-1 (2.93e-2) -	2.1677e-1 (1.43e-5) -	1.4864e-1 (1.76e-2) -	6.6338e-2 (5.94e-3) -	5.3509e-2 (3.01e-3) -	3.0340e-2 (8.46e-4)
SMOP6		3.1261e-1 (7.03e-3) -	3.7539e-2 (7.10e-6) -	1.3509e-1 (3.76e-2) -	6.8198e-2 (7.98e-3) -	6.7243e-2 (7.99e-3) -	3.0327e-2 (7.92e-4)
SMOP7		3.0113e+0 (2.94e-2) -	2.1867e-1 (6.10e-4) -	8.7339e-2 (6.06e-3) +	1.2025e-1 (1.16e-2) -	9.7530e-2 (4.82e-3) =	1.0312e-1 (3.27e-3)
SMOP8		3.7226e+0 (7.49e-3) -	1.3813e+0 (8.49e-3) -	2.2698e-1 (2.94e-2) =	2.5515e-1 (1.35e-2) =	2.9171e-1 (1.48e-2) -	2.3236e-1 (2.43e-2)
+/-/=		1/30/1	0/31/1	1/22/9	0/27/5	0/26/6	

algorithm in population distributivity. It may be related to the fact that our algorithm uses a technique that preserves the boundary solutions [58].

4.5 Computational efficiency analysis

Next, we will compare the six algorithms in terms of running time. The running time comparisons on SMOP1-SMOP8 with two objectives and 800 and 1000 decision variables are shown in Figs. 8 and 9. Our algorithm has the least or less running time on all eight tested functions. The proposed method is substantially faster than MOEAPSL, a sparse algorithm based on neural networks. This is because our actor-critic excavates the nonzero decision variables skillfully, significantly reducing the search space and resulting in faster running times. Furthermore, our algorithm does not require generating many offsprings and evaluating them, which saves time. Algorithms LMEA and MOEADVA also require relatively long running times, which should be because these two algorithms need to consume much time for population evaluation.

4.6 Effectiveness of reinforcement learning for mining sparse variables

Figures 10 and 11 show parallel coordinate plots of the decision variables obtained by the six algorithms on SMOP4 and SMOP5 with 300 decision variables. The sparsity of both problems is set to 0.1, meaning that the 270 decision variables in the Pareto-optimal solution are zero [29]. The decision variables of the Pareto-optimal solution obtained by algorithms MOEADVA and LMEA almost fill the search space, as seen in the two figures. Meanwhile, these two algorithms produce the poorest results in Tables 9–12. Both algorithms are based on decision variable analysis and perform better on separable functions. However, they may not show efficient performance in the face of complex sparse problems due to the restrictive nature of their

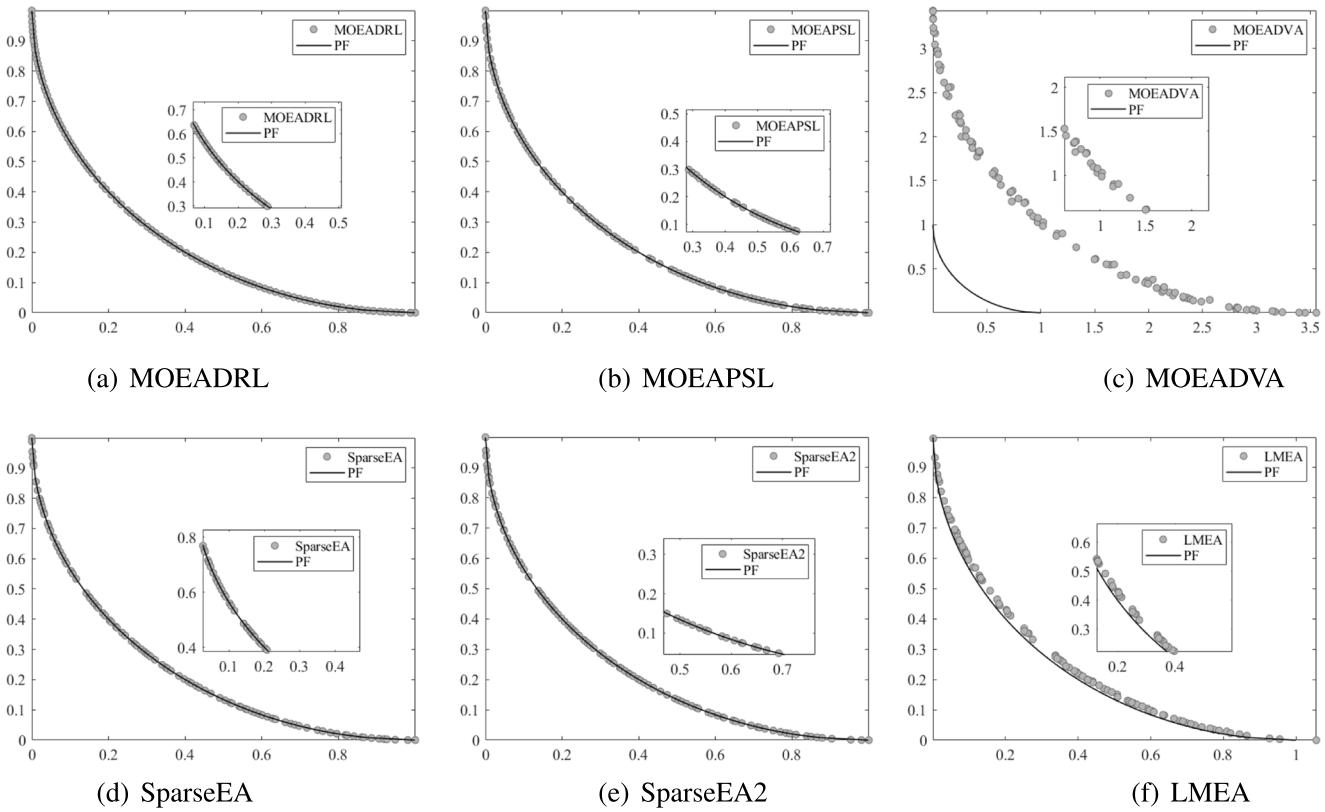


Fig. 6 Pareto optimal fronts with median IGD in 30 runs obtained by six algorithms on 300-dimensional SMOP4 with two objectives

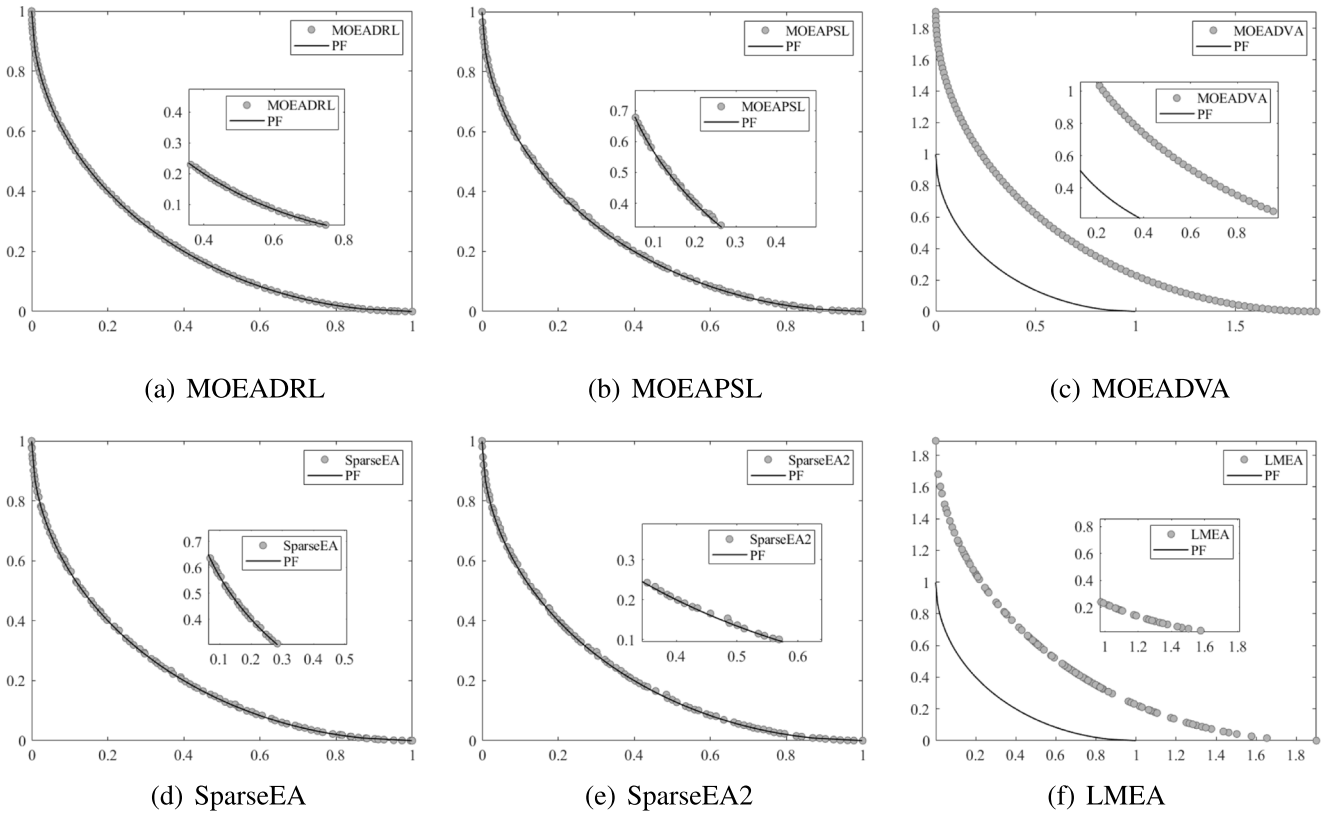


Fig. 7 Pareto optimal fronts with median IGD in 30 runs obtained by six algorithms on 300-dimensional SMOP5 with two objectives

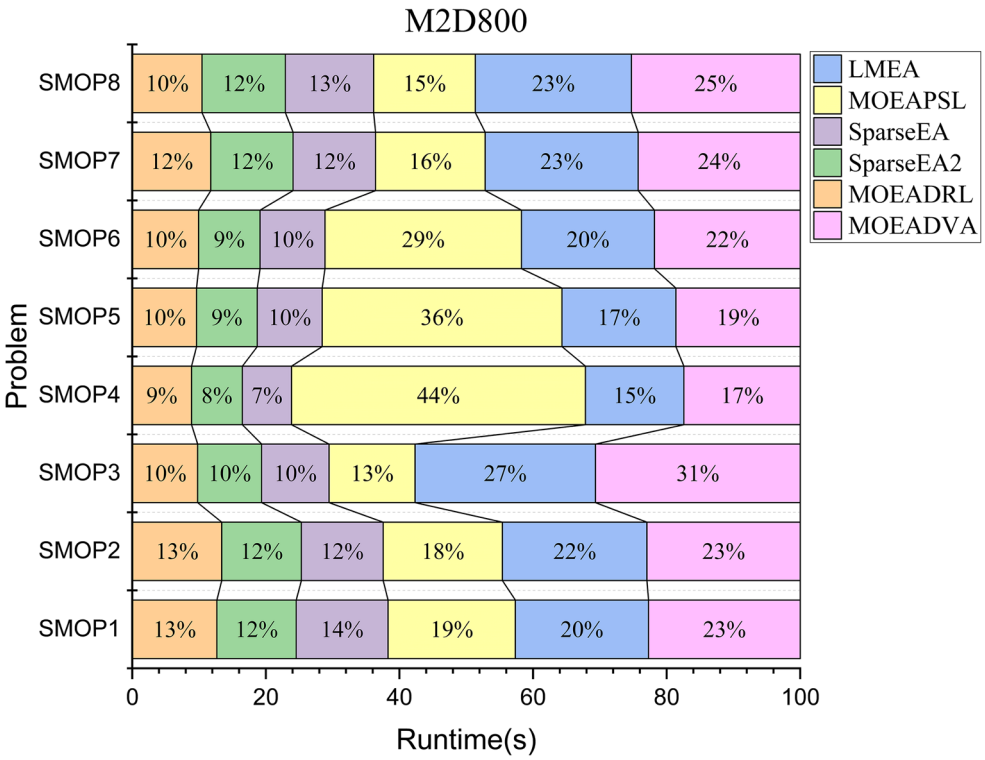


Fig. 8 Runtime comparison of six algorithms on SMOP1-SMOP8 with objective of 2 and decision variable of 800

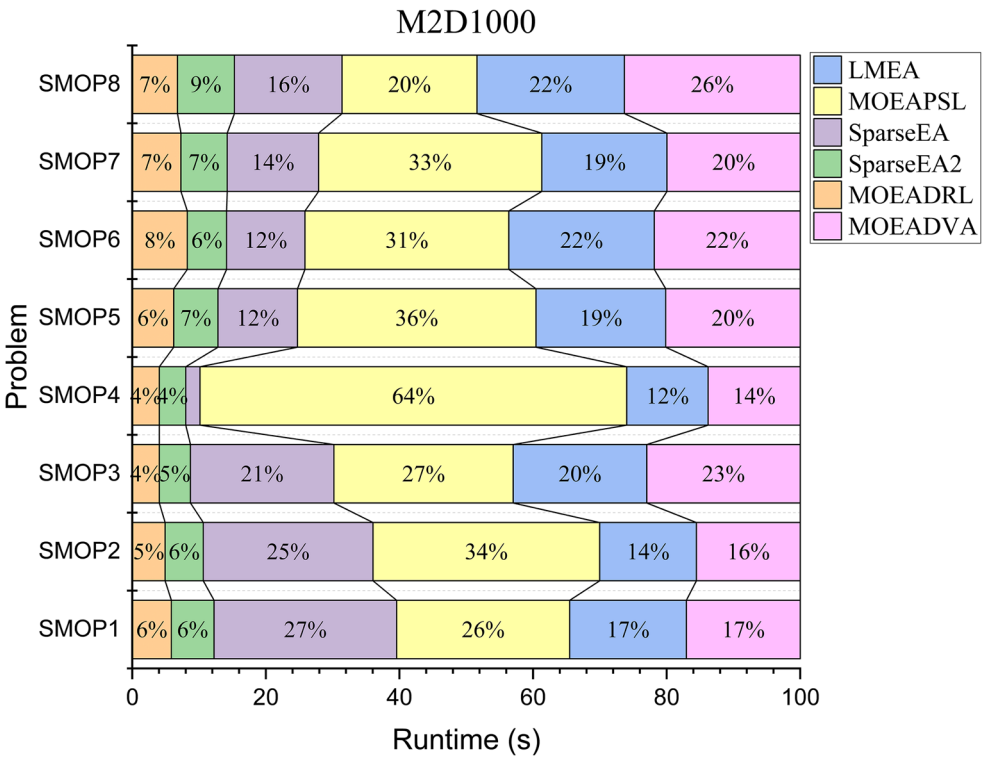
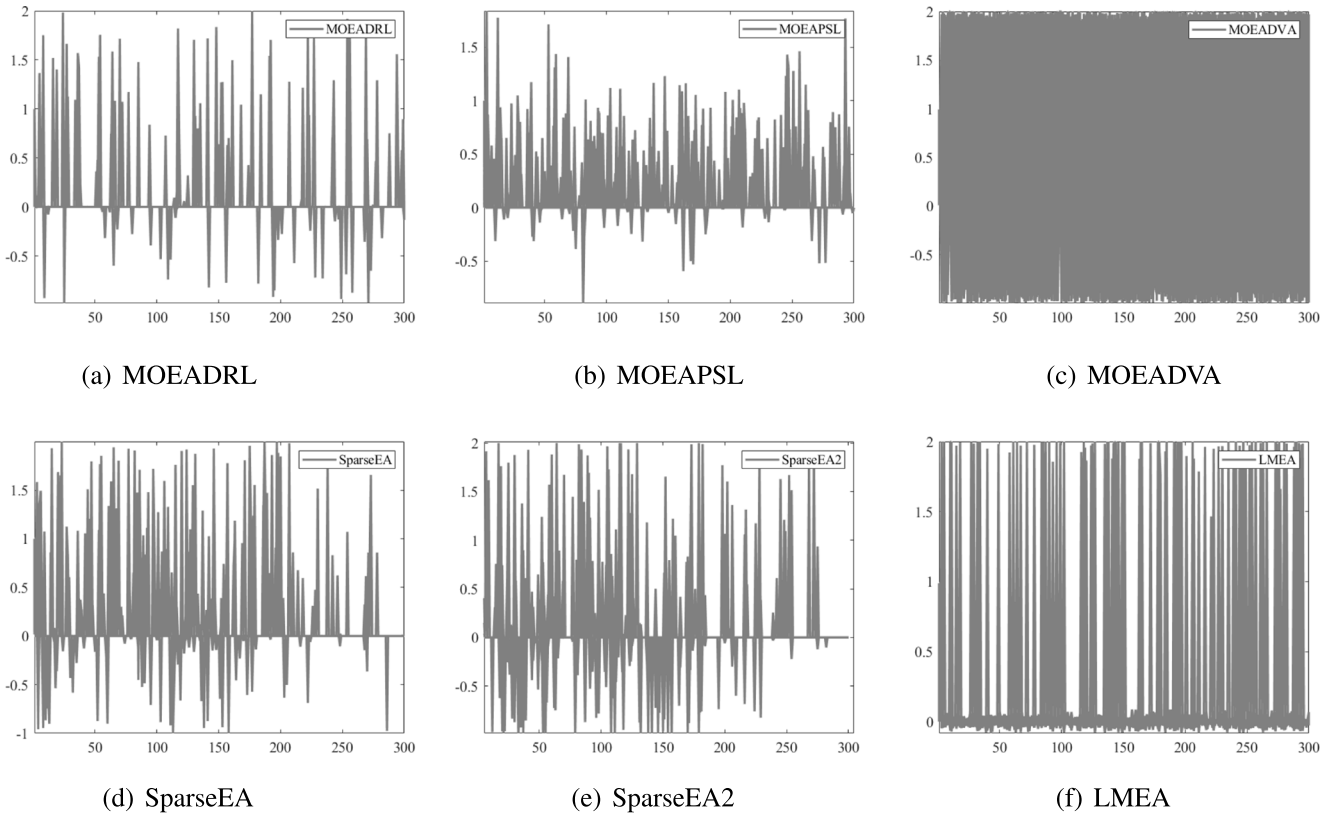


Fig. 9 Runtime comparison of six algorithms on SMOP1-SMOP8 with objective of 2 and decision variable of 1000

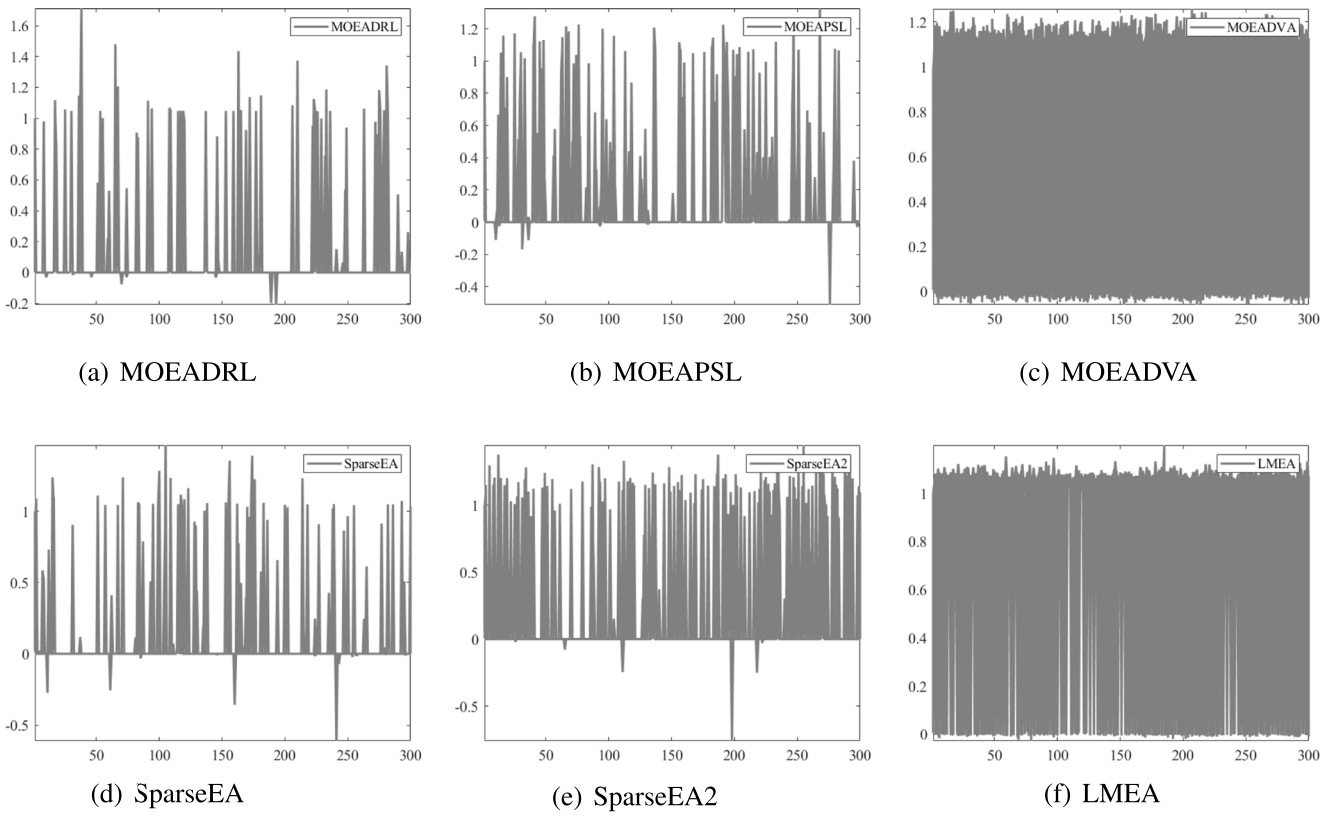
variable grouping. Algorithms MOEAPSL, SparseEA, and SparseEA2, which perform better in other performance comparisons, also have better sparse variable mining ability than algorithms MOEADVA and LMEA in the Pareto-optimal solution comparison. SparseEA2 is an improvement of SparseEA, which is more concerned with optimizing



(a) MOEADRL (b) MOEAPSL (c) MOEADVA

(d) SparseEA (e) SparseEA2 (f) LMEA

Fig. 10 Pareto optimal sets with median IGD in 30 runs obtained by six algorithms on 300-dimensional SMOP4 with two objectives



(a) MOEADRL (b) MOEAPSL (c) MOEADVA

(d) SparseEA (e) SparseEA2 (f) LMEA

Fig. 11 Pareto optimal sets with median IGD in 30 runs obtained by six algorithms on 300-dimensional SMOP5 with two objectives

nonzero variables, so SparseEA2 is inferior to SparseEA in mining sparse variables. The proposed algorithm has the best sparse variable mining ability and obtains a better Pareto front. This fully demonstrates the effectiveness of our reinforcement learning method for mining sparse variables.

4.7 Performance of MOEADRL on Real-World SMOPs

Considering the wide range of sparse problems in academic research and engineering applications, in this subsection, we apply our algorithms to the neural network training problem NN with 321 decision variables, the portfolio optimization problem PO with 1000 decision variables, and the sparse signal reconstruction problem SR with 1024 decision variables [14]. We choose five comparison algorithms to solve these problems, all with the parameters set in 4.1.2. Since the true Pareto fronts are unknown, we use HV values to measure the performance of the algorithms.

Table 13 gives the experimental results of HV values for each algorithm run independently 30 times for all three practical application problems. Bolded shows the optimal results. Our methods yield optimal values for all three real-world application problems, as seen by the results in Table 13. The statistical results also show that our algorithm outperforms the comparison algorithm in most tested instances. This demonstrates the effectiveness and superiority of the proposed algorithm in tackling real-world problems. The efficiency of our algorithms in dealing with large-scale SMOPs can be demonstrated through benchmark problems and practical application experimental results.

5 Conclusion

This paper proposes an efficient evolutionary algorithm based on deep reinforcement learning to solve the large-scale SMOPs, which fully optimizes the nonzero variables while mining and maintaining sparse solutions. The proposed algorithm demonstrates the feasibility of combining reinforcement learning and evolutionary computation to solve large-scale SMOPs. Specifically, we use deep reinforcement learning networks to mine nonzero variables. Then, divide the decision variables into the all-zero group, all-one group, and mixed group based on the idea of the three-way decision. The algorithm efficiency can be improved by delaying the decision for the uncertain variables in the mixed group. In addition, mining and maintaining the all-zero group can highly reduce the search space. Finally, focus on optimizing the decision variables in the all-one group and maintaining the sparsity of the decision variables in the mixed group. Assigning relevant optimization resources for decision variables in different groups can save computational resources.

Table 13 Statistical results of HV values obtained by six algorithms on three application problems with two objectives, the better results are highlighted

Problem	M	D	LMEA	MOEADVA	MOEAPSL	SparseEA	SparseEA2	MOEADRL
NN	2	321	3.3898e-1 (2.05e-2) -	7.7524e-2 (3.33e-3) -	8.9637e-1 (4.39e-3) =	8.9094e-1 (6.30e-3) -	8.9258e-1 (4.06e-3) -	**8.9701e-1 (4.25e-3)**
PO	2	1000	9.1175e-2 (2.80e-5) -	9.1410e-2 (3.73e-5) -	1.1587e-1 (1.01e-2) -	1.2422e-1 (1.12e-3) -	1.2426e-1 (1.25e-3) -	**1.2494e-1 (9.44e-10)**
SR	2	1024	9.1486e-2 (3.11e-3) -	0.0000e+0 (0.00e+0) -	2.5461e-1 (7.69e-2) -	3.3088e-1 (7.95e-2) =	3.2981e-1 (8.00e-2) =	**3.9650e-1 (2.74e-3)**
	+/-/=		0/3/0	0/3/0	0/2/1	0/2/1	0/2/1	

In the experimental study, the proposed algorithm is compared with two MOEAs for LMOPs and three MOEAs tailored for SMOPs on eight benchmark problems and three practical application problems. The statistical results on a total of 67 test instances demonstrate the superiority of the proposed algorithm in solving large-scale SMOPs. Furthermore, we validate the advantages of the algorithm in terms of convergence speed, population diversity, sparse solutions maintenance, and computational efficiency. We experimentally validate the superiority of applying reinforcement learning to solving large-scale SMOPs.

This paper shows that grouping decision variable and assigning corresponding computational resources is practical. However, the utilization of mixed groups in this paper does not reach the highest efficiency, only mining sparse variables in the mixed group. Optimization of the decision variables can be further considered. In addition, this paper uses the well-known environment selection method, and it is necessary to customize an environment selection method for sparse problems. Finally, to the best of our knowledge, there are few studies on large-scale SMOPs, but they all use two-layer coding. We may next attempt a single-layer coding approach to solve large-scale SMOPs directly.

Acknowledgements This work is supported by the National Natural Science Foundation of China (No.62276097), Key Program of National Natural Science Foundation of China (No.62136003), National Key Research and Development Program of China (No. 2020YFB1711700), Special Fund for Information Development of Shanghai Economic and Information Commission (No.XX-XXFZ-02-20-2463) and Scientific Research Program of Shanghai Science and Technology Commission (No.21002411000).

Data availability statement The datasets generated during and analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval This paper does not contain any studies with human participants or animals performed by any of the authors.

Conflict of Interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Ye Tian, Chang Lu, Zhang Xingyi, Cheng Fan, Jin Yaochu (2020) A pattern mining-based evolutionary algorithm for large-scale sparse multiobjective optimization problems. IEEE Trans Cybern, pp 1–14
2. Sarker IH (2021) Machine learning: Algorithms, real-world applications and research directions. SN Comput Sci 2(3):1–21
3. Cope B, Kalantzis M (2022) The cybernetics of learning
4. Gong C, Ren T, Ye M, Liu Q (2021) Maxup: Lightweight adversarial training with data augmentation improves neural network training. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 2474–2483
5. Zhang Q, Ma W, Li G, Ding J, Xie M (2022) Fault diagnosis of power grid based on variational mode decomposition and convolutional neural network. Electr Power Syst Res 208:107871
6. Tan Z, Wang H, Liu S (2021) Multi-stage dimension reduction for expensive sparse multi-objective optimization problems. Neurocomputing 440:159–174
7. Song X-F, Zhang Y, Gong D-W, Sun X-Y (2021) Feature selection using bare-bones particle swarm optimization with mutual information. Pattern Recogn 112:107804
8. Narkhede MV, Bartakke PP, Sutaone MS (2022) A review on weight initialization strategies for neural networks. Artif Intell Rev 55(1):291–322
9. Fan Z, Hu G, Sun X, Wang G, Dong J, Su C (2022) Self-attention neural architecture search for semantic image segmentation. Knowl-Based Syst 239:107968
10. Hashemi A, Dowlatshahi MB, Nezamabadi-pour H (2022) Ensemble of feature selection algorithms: a multi-criteria decision-making approach. Int J Mach Learn Cybern 13(1):49–69
11. Alhenawi E, Al-Sayyed R, Hudaib A, Mirjalili S (2022) Feature selection methods on gene expression microarray data for cancer classification: a systematic review. Comput Biol Med 140:105051
12. Shafiullah Md, Abido MA, Al-Mohammed AH (2022) Intelligent fault diagnosis for distribution grid considering renewable energy intermittency. Neural Comput Applic, pp 1–20
13. Zhang X, Tian Y, Cheng R, Jin Y (2018) A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. IEEE Trans Evol Comput 22(1):97–112
14. Tian Y, Lu C, Zhang X, Tan KC, Jin Y (2020) Solving large-scale multi-objective optimization problems with sparse optimal solutions via unsupervised neural networks. IEEE Transactions on Cybernetics PP(99)
15. Tian Y, Zheng X, Zhang X, Jin Y (2019) Efficient large-scale multi-objective optimization based on a competitive swarm optimizer. IEEE Trans Cybern, pp 1–13
16. Tian Y, Si L, Zhang X, Cheng R, Jin Y (2021) Evolutionary large-scale multi-objective optimization: A survey. ACM Computing Surveys
17. Antonio LM, Coello CAC (2016) Indicator-based cooperative coevolution for multi-objective optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp 991–998
18. Omidvar MN, Yang M, Yi Mei, Li X, Yao X (2017) Dg2: a faster and more accurate differential grouping for large-scale black-box optimization. IEEE Trans Evol Comput 21(6):929–942
19. Sun Y, Yue H (2022) An improved decomposition method for large-scale global optimization: bidirectional-detection differential grouping. Appl Intell 52(10):11569–11591
20. Ma X, Liu F, Qi Y, Wang X, Li L, Jiao L, Yin M, Gong M (2016) A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. IEEE Trans Evol Comput 20(2):275–298
21. He C, Li L, Tian Y, Zhang X, Cheng R, Jin Y, Yao X (2019) Accelerating large-scale multiobjective optimization via problem reformulation. IEEE Trans Evol Comput 23(6):949–961
22. Chen H, Ran C, Wen J, Li H, Jian W (2018) Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations. Inf Sci, p 509
23. 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 Evol Comput 73:101119
24. Tian Y, Zhang X, Wang C, Jin Y (2020) An evolutionary algorithm for large-scale sparse multiobjective optimization problems. IEEE Trans Evol Comput 24(2):380–393

25. Fournier-Viger P, Lin JC-W, Kiran RU, Koh YS, Thomas R (2017) A survey of sequential pattern mining. Data Sci Pattern Recognit 1(1):54–77
26. Alsahaf A, Petkov N, Shenoy V, Azzopardi George (2022) A framework for feature selection through boosting. Expert Syst Appl 187:115895
27. Shetty RD, Bhattacharjee S, Dutta A, Namtirtha A (2022) Gsi: An influential node detection approach in heterogeneous network using covid-19 as use case. IEEE Trans Comput Soc Syst
28. Zhang X, Duan F, Lei Z, Fan C, Jin Y, Ke T (2017) Pattern recommendation in task-oriented applications: a multi-objective perspective [application notes]. IEEE Comput Intell Mag 12(3):43–53
29. Zhang Y, Tian Y, Zhang X (2021) Improved sparseea for sparse large-scale multi-objective optimization problems. Complex Intell Syst, p 10
30. Liu CH, Chen Z, Tang J, Xu J, Piao C (2018) Energy-efficient uav control for effective and fair communication coverage: a deep reinforcement learning approach. IEEE J Sel Areas Commun 36(9):2059–2070
31. Chen L, Jiang S, Liu J, Wang C, Zhang S, Xie C, Liang J, Xiao Y, Song R (2022) Rule mining over knowledge graphs via reinforcement learning. Knowl-Based Syst 242:108371
32. Fan T-H, Wang Y (2022) Soft actor-critic with integer actions. In: 2022 American Control Conference (ACC), pp 2611–2616. IEEE
33. Yuan Y, Lei L, Vu TX, Chatzinotas S, Sun S, Ottersten B (2021) Energy minimization in uav-aided networks: Actor-critic learning for constrained scheduling optimization. IEEE Trans Veh Technol 70(5):5028–5042
34. Wei Y, Yu FR, Song M, Han Z (2019) Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actor-critic deep reinforcement learning. IEEE Int Things J 6(2):2061–2073
35. Liu C-L, Chang C-C, Tseng C-J (2020) Actor-critic deep reinforcement learning for solving job shop scheduling problems. IEEE Access 8:71752–71762
36. Vamvoudakis KG, Lewis FL (2010) Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. Automatica 46(5):878–888
37. Kiumarsi B, Vamvoudakis KG, Modares H, Lewis FL (2017) Optimal and autonomous control using reinforcement learning: a survey. IEEE Trans Neural Netw Learn Syst 29(6):2042–2062
38. Gao M, Feng X, Yu H, Zheng Z (2022) Multi-granularity competition-cooperation optimization algorithm with adaptive parameter configuration. Appl Intell, pp 1–30
39. Schweighofer N, Doya K (2003) Meta-learning in reinforcement learning. Neural Netw 16(1):5–9
40. Peng B, Li X, Gao J, Liu J, Chen Y-N, Wong K-F (2018) Adversarial advantage actor-critic model for task-completion dialogue policy learning. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 6149–6153
41. Zheng Y, Li X, Xu L (2020) Balance control for the first-order inverted pendulum based on the advantage actor-critic algorithm. Inter J Control Auto Syst 18(12):3093–3100
42. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: International conference on machine learning, pp 1928–1937. PMLR
43. Shang K, Ishibuchi H, He L, Pang LM (2021) A survey on the hypervolume indicator in evolutionary multiobjective optimization. IEEE Trans Evol Comput 25(1):1–20
44. Chen H, Dai X, Cai H, Zhang W, Yu Y (2019) Large-scale interactive recommendation with tree-structured policy gradient. In: Proceedings of the AAAI Conference on artificial intelligence, vol 33, pp 3312–3320
45. Zhao S, Liu R, Bo C, Zhao D (2022) Classification-labeled continuousization and multi-domain spatio-temporal fusion for fine-grained urban crime prediction. IEEE Trans Knowl Data Eng, pp 1–14
46. Yang S, Bo Y, Wong H-S, Kang Z (2019) Cooperative traffic signal control using multi-step return and off-policy asynchronous advantage actor-critic graph algorithm. Knowl-Based Syst 183:104855
47. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A et al (2017) Mastering the game of go without human knowledge. Nature 550(7676):354–359
48. Zhang B, Hu W, Cao D, Li T, Zhang Z, Chen Z, Blaabjerg F (2021) Soft actor-critic–based multi-objective optimized energy conversion and management strategy for integrated energy systems with renewable energy. Energy Convers Manag 243:114381
49. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M et al (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489
50. Memarian F, Goo W, Lioutikov R, Niekum S, Topcu U (2021) Self-supervised online reward shaping in sparse-reward environments. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 2369–2375
51. Zhan J, Ye J, Ding W, Liu P (2021) A novel three-way decision model based on utility theory in incomplete fuzzy decision systems. IEEE Trans Fuzzy Syst
52. Yao Y (2021) The geometry of three-way decision. Appl Intell 51(9):6298–6325
53. Bo Y, Li J (2020) Complex network analysis of three-way decision researches. Int J Mach Learn Cybern 11(5):973–987
54. Yang X, Li T, Tan A (2020) Three-way decisions in fuzzy incomplete information systems. Int J Mach Learn Cybern 11(3):667–674
55. Li H, Zhang L, Huang B, Zhou X (2016) Sequential three-way decision and granulation for cost-sensitive face recognition. Knowl-Based Syst 91:241–251
56. Zhang Q, Pang G, Wang G (2020) A novel sequential three-way decisions model based on penalty function. Knowl-Based Syst 192:105350
57. Ma Y, Bai Y (2020) A multi-population differential evolution with best-random mutation strategy for large-scale global optimization. Appl Intell 50(5):1510–1526
58. Wang H, Jiao L, Yao X (2015) Twoarch2: an improved two-archive algorithm for many-objective optimization. IEEE Trans Evol Comput 19(4):524–541
59. Tian Y, Cheng R, Zhang X, Jin Y (2017) Platemo: a matlab platform for evolutionary multi-objective optimization [educational forum]. IEEE Comput Intell Mag 12(4):73–87
60. Ishibuchi H, Imada R, Setoguchi Y, Nojima Y (2018) Reference point specification in inverted generational distance for triangular linear pareto front. IEEE Trans Evol Comput 22(6):961–975
61. Said R, Bechikh S, Louati A, Aldaej A, Said LB (2020) Solving combinatorial multi-objective bi-level optimization problems using multiple populations and migration schemes. IEEE Access 8:141674–141695

Mengqi Gao received the M.Sc. degree in Computer Science and Computing from Zhejiang Normal University, China, in 2019. She is currently a doctoral in Computer Science, East China University of Science and Technology. Her research interests include Reinforcement learning, swarm intelligence and evolutionary computing, big data intelligence.

Xiang Feng received the M.Sc. degree in System Engineering from Naval Engineering University, China, in 2003. She received the Ph.D. degree in Control Theory and Engineering from East China University of Science and Technology, China, in 2006. She worked as a postdoctoral fellow in the Department of Computer Science of the University of Hong Kong from 2006 to 2008. She is presently a professor of the Department of Computer Science and Engineering, East China University of Science and Technology. Her research interests include distributed swarm intelligence and evolutionary computing, integration learning and integration optimization, big data intelligence.

Huiqun Yu received the M.Sc. and Ph.D. degrees in Computer Science from the East China University of Science and Technology and Shanghai Jiaotong University, in 1992 and 1995, respectively. He is presently a professor of the Department of Computer Science and Engineering, East China University of Science and Technology. His main research interests include software engineering, trustworthy computing and big data intelligence.

Xiuquan Li received an M.Sc. and Ph.D. degrees in Computer Science from the East China University of Science and Technology and Tsinghua University respectively. He is presently a researcher at the China Academy of Science and Technology Development Strategy and a deputy director of the New Generation Artificial Intelligence Development Research Center of the Ministry of Science and Technology. His main research interests include big data and artificial intelligence technology prediction and evaluation, industrial technology roadmap, policy research.