Energy Conversion and Management 297 (2023) 117637

Contents lists available at ScienceDirect

Energy Conversion and Management

journal homepage: www.elsevier.com/locate/enconman

Multi-objective optimization of turbine blade profiles based on multi-agent reinforcement learning

Lele Li [a,b], Weihao Zhang [a,b,*], Ya Li [c], Chiju Jiang [a,b], Yufan Wang [a,b]

[a] School of Energy and Power Engineering, Beihang University, Beijing 100191, China
[b] National Key Laboratory of Science and Technology on Aero-Engine Aero-thermodynamics, Beijing 100191, China
[c] School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876, China

ARTICLE INFO

Keywords:
Aerodynamic design
Dynamic multi-objective optimization
Reinforcement learning
Rapid optimization

ABSTRACT

The aerodynamic design level of the blade profile directly affects the overall energy conversion efficiency of the turbine. However, the optimization process of the blade profile is a typical multi-objective and multi-constraint optimization problem. Traditional optimization algorithms tend to fall into local optima and have slow solving speeds when dealing with these types of problems. To address these issues, this study proposes a dynamic multi-objective optimization algorithm based on multi-agent reinforcement learning (DMORL). This algorithm describes the aerodynamic performance optimization process of the blade as a Markov decision process and employs a multi-agent collaborative optimization strategy to parallelize the solution for different optimization objectives. After the model training is completed, it can provide the Pareto front in real time under different geometric constraints and airflow incidence angles, accomplishing dynamic multi-objective optimization of the blade profile. Experimental results demonstrate that, compared to traditional multi-objective optimization algorithm (NSGA-II), DMORL can find a better Pareto front, with an average solving time of only 0.12 s per multi-objective optimization problem, improving optimization speed by 51 times.

1. Introduction

With the continuous growth in global energy demand and the increasing depletion of traditional energy resources, finding methods to utilize energy efficiently has become critically important. Turbines serve as a vital energy conversion device and have been extensively employed in diverse fields such as power generation [1,2], aviation [3,4], and maritime applications [5–7]. Their performance directly influences the conversion efficiency and utilization of energy. The aerodynamic design and optimization of turbine blade profiles constitute a crucial aspect of enhancing turbine efficiency. In early studies, experiments were the primary method for obtaining the aerodynamic performance of turbine blades. However, when we use experimental methods for the aerodynamic design of turbine blade profiles, it may require hundreds or even thousands of measurements on the blade surface pressure distribution to estimate the aerodynamic efficiency of the blade. In recent years, with the development of Computational Fluid Dynamics (CFD), turbine blade design based on CFD has become increasingly favored by researchers. This method significantly reduces the cost of aerodynamic design for turbine blades. Nonetheless, design and optimization based on CFD often require a large number of CFD simulations. When computational resources are insufficient, this process can severely slow down the design cycle for the blade.

In order to solve the above problems, blade design and optimization methods based on surrogate models have gradually developed. Sanga Lee et al. [8] established a mapping relationship between rotor blade profile and turbine stage efficiency using the Kriging model, and embedded it into the optimization algorithm, the optimized turbine efficiency improved by 1.63 %. Sang Lae et al. [9] used polynomial response surface method to optimize wind turbine blades. Under the condition of unchanged thrust, the efficiency of the optimized blade profile increased by about 10 %. Dewei Tang et al. [10] employed an Artificial Neural Network (ANN) to establish a connection between rotorcraft blade profile parameters and power. Compared with conventional blade, the optimized blades increased power loading by 24 %. Yuqi Wang et al. [11] utilized Convolutional Neural Networks (CNN) to predict the pressure and temperature distribution on the blade surface, with prediction errors consistently within 1 %.

After the surrogate model is established, the final result of blade profile optimization depends on the choice of optimization algorithm. Currently, the optimization strategies applied to blade design are mainly

* Corresponding author at: School of Energy and Power Engineering, Beihang University, Beijing 100191, China.
E-mail address: zhangweihao@buaa.edu.cn (W. Zhang).

https://doi.org/10.1016/j.enconman.2023.117637
Received 17 July 2023; Received in revised form 26 August 2023; Accepted 5 September 2023
Available online 11 October 2023

Nomenclature

A	Action set of the agent
C_x	Blade Axial chord
$C_p^{'}$	Optimal C_p values in an episode
C_p^{best}	Optimal C_p values for neural network prediction
Ma	Mach number
o	Blade throat length
p	Static pressure
Q_π	Action-value function
R	Reward function or radius of the blade leading or trailing edge
S	State set of the agent
U	Discounted return
w	Parameters of the value network
Z_w	Zweifel coefficient
α	Incidence angle or learning rate
β_{sa}	Stagger angle
Θ	Weights and biases of U-Net
κ	Weight coefficients of the reward function
Ω_{obs}	The results of CFD simulations
π	Policy function
a	Action of the agent
C_p	Total pressure loss coefficient
C_p^*	Optimal C_p value in all episodes
$\mathbb{E}$	Expectation
N	Number of blades
P	State transition probability
p^*	Total pressure
Q^*	Optimal action-value function
r	Reward of the agent
s	State of the agent
V_π	State-value function
y^+	Non-dimensional thickness of the first layer of meshes
Z_w^{best}	Optimal Z_w values for neural network prediction
β	Inlet or outlet blade angle or learning rate
γ	Discount factor or blade wedge angle or ratio of specific heats
θ	Parameters of the policy network
ξ	Blade unguided turning
Ω_{model}	Predictions of U-Net
$\mathscr{B}$	Replay buffer

Subscripts

le	Blade leading edge
t	Time step
in	Inlet geometry of the blade profile
opt	Optimal
2	Outlet of the computational domain or U-Net2
te	Blade trailing edge
t_{max}	Maximum time step
out	Outlet geometry of the blade profile
1	Inlet of the computational domain or U-Net1

Superscripts

e	Extrinsic reward
int	Interactive reward
2	Agent 2
i	Intrinsic reward
1	Agent 1

divided into two categories: gradient-based optimization methods and stochastic optimization methods. Gradient-based optimization methods are usually combined with CFD calculation process to solve optimal aerodynamic performance parameters using adjoint matrices. Haitao Li et al. [12] proposed an adjoint method independent of the Navier-Stokes solver and applied it to the aerodynamic design of turbine blades. The total pressure coefficient of the blades increased by about 3 %. Jiqi Luo et al. [13] applied the gradient-based adjoint method to the three-dimensional design of turbine blades. Under certain constraints, the isentropic efficiency of the optimized blades increased by about 0.6 %. Gradient-based optimization methods require the calculation of the Jacobian or Hessian matrix during the optimization process, which can result in significant computational overhead. Additionally, this method has certain requirements for the initial solution, and the algorithm may not converge when the initial solution is chosen inappropriately.

Stochastic optimization methods differ greatly from gradient-based optimization methods. This method typically does not require the calculation of gradient information during the solving process and imposes almost no additional requirements on the initial solution. Therefore, compared to gradient-based optimization algorithms, this method has better generalization and wider applicability. C.M. Chan et al. [14] combined the genetic algorithm (GA) with CFD to aerodynamically design wind turbine blades. After optimization, the time-averaged power coefficient increased by 33 %. Long Wang et al. [15] improved the nondominated sorting genetic algorithm (NSGA-II) and applied it to the design of 5 MW wind turbine blades. Similar works include [16–18]. C.C. Liao et al. [19] proposed an improved particle swarm optimization algorithm (PSO) for the weight reduction design of wind turbine blades. After 50 iterations (taking approximately 25 h), the blade mass decreased by 2.3 %. Yingjue Li et al. [20] developed a multi-disciplinary coupled optimization design method based on a multi-objective particle swarm optimization algorithm (MOPSO). The simulation results showed a 6.9 % increase in the blade power coefficient and a 3.1 dB reduction in noise after optimization. Ramazan Özkan et al. [21] used the artificial bee colony algorithm (ABC) to optimize the design of small-scale wind turbine blades. The experimental results show that the ABC method significantly improves the aerodynamic performance and power of the blades. In addition to the above algorithms, ant colony algorithm and simulated annealing algorithm have also been applied in blade optimization design [22–25].

Although the stochastic optimization method has strong robustness, it is prone to getting trapped in local optima for high-dimensional and strongly nonlinear problems. Additionally, current algorithms for solving multi-objective optimization problems often convert them into single-objective optimization problems through weighted linear combinations, but this method is difficult to assign weight coefficients to each objective [26]. When using multi-objective optimization algorithms such as NSGA-II and MOPSO to solve these problems, the optimization efficiency of the algorithm will be greatly reduced. In addition to the above-mentioned shortcomings, traditional algorithms also suffer from the problem of difficulty in reusing historical optimization results. For example, after optimizing a certain optimization problem, if some conditions of the original optimization problem change, we can only obtain the optimal solution of the new optimization problem by executing the optimization algorithm again. This is mainly because the optimization results obtained by traditional optimization algorithms in one optimization are limited in guiding similar optimization problems.

Reinforcement learning (RL) is an important branch of machine learning. It focuses on enabling agents to learn optimal strategies through interacting with the environment to achieve specific goals. In recent years, with the development of RL, RL has been gradually applied to optimization problems. For example, Liangyue Jia et al. [27]

presented a RL-based method for searching for the optimal twist angle distribution of wind turbine blades efficiently. Zheng Wang et al. [28] proposed a wind turbine optimization method based on the deep deterministic policy gradient (DDPG) algorithm, which reduces the optimization time by 75.52 % compared with the state-of-art models. Jianhao Fang et al. [29] endeavored to find the optimal rotor speed of wind turbines under different rainfall intensities and wind speeds using RL method. The optimized speed can extend the surface coating life of wind turbine blades by 2.25 times, while the energy yield only decreases by 0.025 %. Zhiwen Deng et al. [30] proposed a decentralized yaw optimization method based on RL for maximizing wind farm power output. Guozhou Zhang et al. [31] designed a multi-energy hub system driven by renewable energy and utilized RL algorithms to optimize the system. The study showed this method reduced the total cost by 7.28 %. Bin Zhang et al. [32] applied RL to energy conversion and management. Simulation results demonstrated this approach effectively improved the system operator's profit. Other similar research can be found in References [33,34]. From these research results, RL shows advantages in faster optimization speed and reusable historical optimization results compared to traditional optimization algorithms. After solving an optimization problem, it can instantly infer the optimal solution for similar problems without the need to repeat the optimization process.

The previous research on RL-based optimization methods were based on single-agent systems. Single-agent reinforcement learning (SARL) has advantages in simple low-dimensional problems, but in high-dimensional, strongly nonlinear problems, its performance is inferior to that of multi-agent reinforcement learning (MARL) [35]. MARL assigns different optimization problems to different agents that collaborate with each other to find all solutions to the multi-objective optimization problem [36]. As the optimal solution to different optimization objectives is completed by different agents, it is easy to solve in parallel and has high optimization efficiency. Although MARL has significant advantages compared to SARL, research in this field is still in its infancy and there remain many issues that need to be addressed. To fill this gap, this study proposes a dynamic multi-objective optimization method for turbine blade profiles based on multi-agent reinforcement learning (DMORL). This research first systematically describes the modeling process of MARL for the multi-objective design and optimization problem of blades. Then, two types of reward functions are specifically designed for the turbine blade optimization problem within the MARL framework. Finally, to verify the performance of DMORL, it is compared with the most widely applied multi-objective optimization algorithm currently. Compared with previous research, the main contributions of this paper are as follows:

(1) A high-precision prediction model of blade aerodynamic performance based on U-Net was established. The prediction error of this model for various aerodynamic parameters is within 5 %.
(2) A multi-objective dynamic optimization algorithm based on reinforcement learning is proposed. After training, this algorithm can optimize the blade profile in real time under changing working conditions and constraints.
(3) Two new multi-agent reinforcement learning reward design schemes are presented. In solving dynamic multi-objective optimization problems, these schemes significantly improve the optimization effect.
(4) The DMORL framework proposed in this paper is the first attempt to use multi-agent reinforcement learning to solve multi-objective optimization problems in turbine blade design and the reliability of this method has been verified through CFD experiments.

The paper's structure is organized as follows: Section 2 provides the theoretical background; Section 3 delineates a multi-objective optimization method based on reinforcement learning; Section 4 conducts an analysis of the experimental results; Lastly, Section 5 furnishes a comprehensive summary of the entire paper.

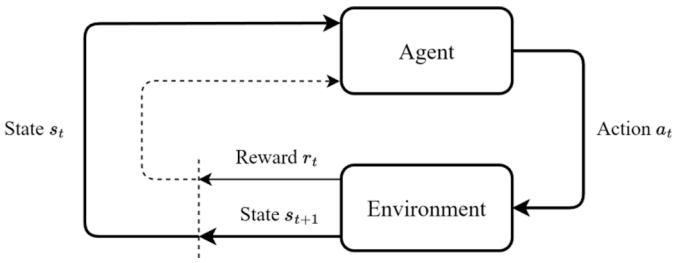


Fig. 1. The fundamental framework of reinforcement learning.

2. Theoretical background

2.1. Reinforcement learning

RL is different from supervised or unsupervised learning problems in machine learning. The goal of RL is to learn a policy through interaction with the environment to maximize cumulative rewards. Fig. 1 shows the basic framework of RL. RL mainly consists of five components: 1) agent, 2) environment, 3) action, 4) state, and 5) reward. RL problems are defined on Markov decision processes (MDP). An MDP is a five-tuple as (S,A,P,R,γ). S is the set of all states, where $s_t \in S$ represents the state that the agent is in at time t; A represents the action set, and $a_t \in A$ represents the action that the agent takes at time t; P is the state transition probability, where $P(s^{'}|s,a)$ represents the probability of transitioning from state s to state $s^{'}$ after taking action a; R represents the reward function. The reward r_t at time t is determined by the state s_t and action a_t; γ is the discount factor, that determines the agent's allocation of weights between short-term returns and long-term returns.

Fig. 1 describes the interactive learning process between the agent and the environment in RL. At time t, the agent obtains the environment state s_t, calculates the action a_t and then executes the action a_t. The environment returns the reward r_t at time t and the state s_{t+1} at time $t+1$. This process continues until the agent achieves its goal or reaches the maximum time step. The discounted return of the agent at time t is denoted as:

$$U_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots \gamma^n r_{t+n} \quad \gamma \in [0,1] \tag{1}$$

U_t represents the total sum of weighted rewards starting from time t. In the above formula, γ determines the rate of decay of future rewards. As γ approaches 1, the agent focuses more on long-term returns, while when γ approaches 0, the agent prioritizes short-term returns. The agent's action at each time step is controlled by the policy function π, and this process can be described by the following equation.

$$a_t = \pi_\theta(s_t) \tag{2}$$

Here, θ represents the parameters of the policy function. The goal of RL is to find the optimal policy function that maximizes the cumulative reward. The action-value function is defined as follows:

$$Q_\pi(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t] \tag{3}$$

Q_π represents the effectiveness of taking action a_t given policy function π and state s_t. Based on the above equation, the optimal action-value function is defined as follows:

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t) \tag{4}$$

Q^* represents the value obtained by choosing the policy that maximizes the Q_π value among all policies. According to Q^*, we can choose the optimal action a_t given s_t. Define the state-value function:

$$V_\pi(s_t) = \mathbb{E}_A[Q_\pi(s_t, A)] \tag{5}$$

V_π represents the expected value obtained by performing all possible

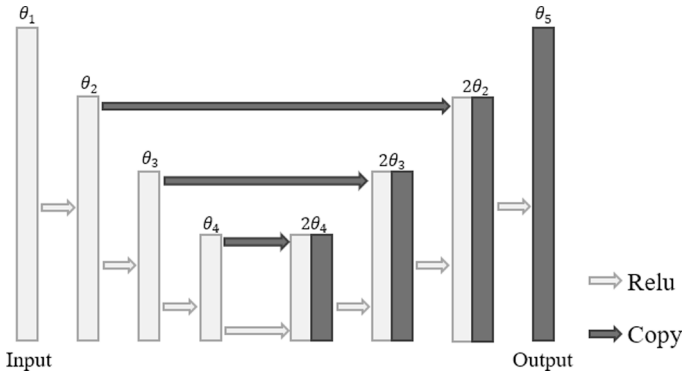


Fig. 2. U-Net network framework.

actions in state s_t, and is only dependent on the current state s_t and policy π. It evaluates the goodness of the current situation.

The method of obtaining the optimal policy π based on Q^* is called policy learning, such as Policy Gradient (PG) [37], Actor-Critic (A2C) [38], and Trust Region Policy Optimization (TRPO) [39]. The method of obtaining the optimal policy π based on V_π is called value learning, such as Q-Learning [40], Deep Q-Network (DQN) [41] and Dueling DQN [42].

2.2. Dynamic multi-objective optimization

Dynamic multi-objective optimization problems (DMOPs) often involve multiple conflicting objective functions, and these objective functions may change during the optimization process. Traditional algorithms are often insufficient for solving such problems [43]. The DMOPs is defined as follows:

$$\begin{aligned} &\text{Minimize} \quad f(x,t) = [f_1(x,t), f_2(x,t), \ldots, f_M(x,t)] \\ &\text{subject to} \quad x \in \Omega \end{aligned} \tag{6}$$

In the above formula, $f(x,t)$ represents a vector of M objectives, which may change continuously with time t, $x = (x_1, x_2, \ldots, x_n)$ is a discrete vector, and Ω represents the domain of the independent variable x. The Pareto optimal x^* is defined as the set of solutions (DPOS) at time t that are not dominated by any other solutions x:

$$DPOS = \{x^* \mid \neq g \exists x \in \Omega : x \prec_t x^*\} \tag{7}$$

The dynamic Pareto front (DPOF) is defined as the set of DPOS for all objectives at time t:

$$DPOF = \{f(x^*,t) = (f_1(x^*,t), f_2(x^*,t), \ldots, f_M(x^*,t)) \mid x^* \in DPOS\} \tag{8}$$

Previous research has typically employed enhanced stochastic optimization algorithms to solve DMOPs. Such as Ruochen Liu et al. [44] proposed an improved PSO algorithm that can solve DMOP through multiple particle swarms. Radhia Azzouz et al. [45] applied evolutionary algorithms to solving DMOP problems. Similar works also include [46–48], refer to [43]. These research works provide solutions to DMOPs, but the algorithms they developed are often more complex than traditional multi-objective optimization algorithms. This makes it difficult for such methods to respond in real time when optimization objectives or constraints change.

2.3. U-Net network

U-Net was proposed by Ronneberger et al [49]. This network was originally used for medical image segmentation tasks, but due to the superiority of the network structure, more and more researchers have applied it to other fields [50–52].

The architecture of the U-Net network is shown in Fig. 2, which consists of two parts: the encoder and the decoder. The encoder progressively reduces spatial information while increasing contextual information. Conversely, the decoder gradually restores spatial and detail information. The U-Net employs special skip connections between layers, directly linking the feature maps from the encoder to the corresponding layers of the decoder, thereby realizing an end-to-end training process. Additionally, due to its relatively simple structure, the U-Net model offers fast inference speed and strong generalizability.

To shorten the blade profile optimization cycle, a surrogate model is established using the U-Net network in this paper. The model structure is shown in Fig. 3, which achieves a nonlinear mapping from geometric parameters to the total pressure loss coefficient and Zweifel coefficient of the turbine blade. The total pressure loss coefficient is defined as follows:

$$C_p = \frac{p_1^* - p_2^*}{p_2^* - p_2} \tag{9}$$

Where p^* is the total pressure, p is the static pressure, and subscripts 1

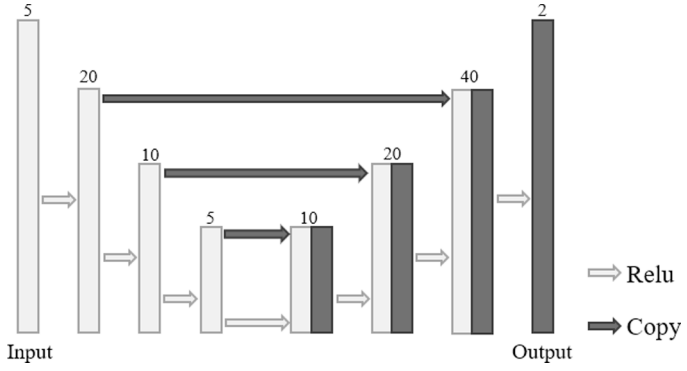


Fig. 3. U-Net network detailed framework.

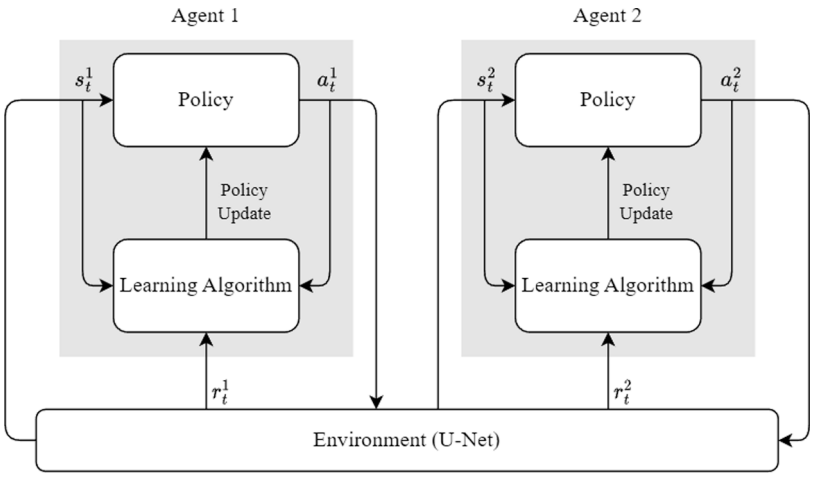


Fig. 4. DMOP framework based on MARL.

and 2 represent the inlet and outlet of the turbine blade, respectively. C_p is one of the most commonly used indicators in the initial design stage of blade profile, which measures the level of aerodynamic performance of the blade profile. The definition of the Zweifel coefficient is as follows:

$$Z_w = \oint \frac{p}{p_1^* - p_2} \mathrm{d}\left(\frac{x}{b_x}\right) \tag{10}$$

In the above equation, x represents the axial position (coordinate) along the blade profile, and b_x is the axial chord of the blade. Z_w measures the magnitude of the load on the blade and represents the ratio of the circumferential force acting on the blade to a certain ideal circumferential force. This ideal circumferential force is the circumferential force acting on the blade when the static pressure on the pressure surface is uniform and equal to the inlet total pressure, and the static pressure on the suction surface is uniform and equal to the outlet static pressure [53]. For U-Net, it implements the mapping relationship in Eq. (11).

$$\Omega = F(\Theta, X) \tag{11}$$

As shown in the Eq. (11), Θ represents the parameters of the network, $\Theta = (w, b)$, where w and b represent the weights and biases of U-Net respectively. X is the input of the U-Net, specifically in this paper, it refers to the geometric parameters of the blade (see Section 3.2 for details). Ω represents the output of U-Net, which is a vector composed of C_p and Z_w.

3. Methodology

3.1. Dynamic multi-objective optimization of turbine blade profiles based on MARL

To address the DMOPs in blade's shape optimization, this paper employs MARL for the optimization process. In a MARL system, agents can communicate with each other to achieve policy complementarity and enhance learning efficiency. Additionally, the agents in MARL can collaborate with each other to accomplish collaborative tasks or compete with each other for cooperative evolution. This makes MARL capable of addressing even more complex problems. Due to these advantages, MARL is currently widely used in applications such as autonomous driving [54,55] and distributed control [56,57]. For the DMOPs studied in this research, MARL demonstrates stronger adaptive capabilities in dynamic environments compared to previous dynamic optimization algorithms [44–47]. More importantly, DMOPs often require real-time decision making, which MARL can accomplish through online learning. This learning approach not only saves computational resources but also allows the model to continuously improve its own performance, enhancing optimization efficiency.

The dynamic multi-objective optimization framework based on MARL in this study is illustrated in Fig. 4. The optimization problem defined in this paper is as follows:

$$\begin{aligned} &\text{Minimize} \quad f(x,t) = [f_1(x,t), f_2(x,t)] \\ &\text{subject to} \quad x \in \Omega \end{aligned} \tag{12}$$

Where f_1 represents the C_p of the blade, f_2 is the negative of the $-Z_w$, and x is the geometric parameters. This objective function means finding a turbine blade profile with the best aerodynamic performance and the maximum load within the given geometric constraint range.

To solve the above problem, this study designed two agents to handle them separately (corresponding to agent 1 and agent 2 in Fig. 4). Agent 1 is mainly responsible for optimizing the C_p of the blade, while agent 2 is responsible for optimizing the Z_w. The multi-agent MDP can be formulated as $(S^1, S^2, A^1, A^2, P, R^1, R^2, \gamma)$, where the superscripts 1 and 2 represent agent 1 and agent 2, respectively. At time step t, the agent's state is defined as:

$$S_t = \left[x_t^1, x_t^2\right] \tag{13}$$

Based on the state S_t and policy function π, the action at time step t is given by:

$$\begin{cases} A_t = \left[\Delta x_t^1, \Delta x_t^2\right] \\ \Delta x_t^1 = \pi^1\left(x_t^1\right) \\ \Delta x_t^2 = \pi^2\left(x_t^2\right) \end{cases} \tag{14}$$

x_t^1 and x_t^2 respectively represent the geometric parameters of the blade profile for agent 1 and agent 2 at time step t, while Δx_t^1 and Δx_t^2 represent the change in the blade profile parameters at time step t. The corresponding reward under this state is provided by the environment, which is constructed by a U-Net network.

$$\begin{cases} r_t^1 = F\left(\Theta, x_t^1\right) \\ r_t^2 = F\left(\Theta, x_t^2\right) \end{cases} \tag{15}$$

In the above formula, $r_t^1 = f_1, r_t^2 = f_2$. The state of the environment at time step $t+1$ is S_{t+1}:

$$\begin{cases} S_{t+1} = \left[x_{t+1}^1, x_{t+1}^2\right] \\ x_{t+1}^1 = x_t^1 + \Delta x_t^1 \\ x_{t+1}^2 = x_t^2 + \Delta x_t^2 \end{cases} \tag{16}$$

The aforementioned process continues until the maximum time step $t_{\max}$ is reached. In RL, an episode is defined as:

$$\left[(S_0, A_0, r_0), (S_1, A_1, r_2) \cdots (S_{t_{\max}}, A_{t_{\max}}, r_{t_{\max}})\right] \tag{17}$$

Table 1
Design blade profile parameters and ranges.

Parameter	Range	Parameter	Range
β_{sa}	56.31°–85.41°	ξ	10°
R_{le}/C_x	0.27×10^{-1} –0.67×10^{-1}	β_{in}	−39°
γ_{in}	20°–50°	β_{out}	−59°
R_{te}/C_x	0.17×10^{-1} –0.33×10^{-1}	N	70
γ_{out}	5°–20°	o/C_x	0.38
C_x	15 mm	–	–

An episode represents one complete interaction between the agents and the environment. The goal of RL is to obtain an optimal policy function $\pi^*\left(\pi^1_{opt},\pi^2_{opt}\right)$, which can guide the agents to take the optimal action at each time step to maximize the cumulative reward. The update of policy function parameters is carried out by the RL learning algorithm.

Taking into account the geometric constraints and variations in the working conditions during the design process, the optimization problem discussed in this paper can be reformulated as a constraint-handling DMOPs.

$$\begin{aligned}&\text{Minimize} \quad f(\dot{x},\alpha,x,t)=[f_1(\dot{x},\alpha,x,t),f_2(\dot{x},\alpha,x,t)]\\&\text{subject to} \quad x\in\Omega,\ \ \dot{x}=x_{\text{give}},\ \ \alpha=\alpha_{\text{give}}\end{aligned} \tag{18}$$

In the above equation, $\dot{x}$ and α represent the given geometric constraints and airflow incidence angle, respectively. When using MARL to solve this problem, the definitions of state, action, and reward are the same as Eq. (13), (14), and (15). The difference lies in solving the unconstrained DMOPs, where the initial state x of each episode is random. However, when solving constraint-handling DMOPs, the initial state x of the unconstrained parameter is random while the constrained parameters $\dot{x}$ and α are given (see Section 4.3 for details). The following sections provide a detailed description of all the components of this optimization framework.

3.2. Environment

The environment is an important component in RL. All the processes in RL are performed within a given environment. The environment provides feedback for every action taken by the agent, which serves as the driving force in RL, guiding the agent to adjust its strategy to maximize cumulative rewards. In the blade profile optimization problem studied in this paper, the environment is an evaluation system of blade profile aerodynamic performance and load level. This system can be constructed by CFD solvers or consist of surrogate models (such as response surface models or empirical models). To improve optimization efficiency, this study employs the U-Net network mentioned in Section 2.3 to establish an RL environment suitable for the multi-objective optimization problem of turbine blade profiles.

Constructing a RL environment based on U-Net consists of the following steps: 1) generating a blade profile dataset, 2) conducting CFD calculations, 3) model training and testing. This paper employs an improved eleven-parameter method to generate the blade profile data [58]. The method uses Bezier curves to generate the pressure and suction surfaces of the blade profile, ensuring that the profile curvature remains continuous. The input parameters for the eleven-parameter modeling method are shown in Table 1. Previous studies [59,60] have indicated that the installation angle (β_{sa}), leading edge radius (r_{le}), leading edge wedge angle (γ_{in}), trailing edge radius (r_{te}), and trailing edge wedge angle (γ_{out}) have a significant impact on the aerodynamic performance and load level of the blade profile. Therefore, this study changes only these five geometric parameters when generating the data set. Their detailed value ranges are displayed in Table 1.

After obtaining the blade profile data, the NUMECA AutoGrid5 is utilized to mesh the turbine blades. An O-type grid is adopted around the blade profile, while an H-type grid is used for the remaining areas. The y^+ near the wall is set to be approximately 1 to ensure that the turbulence model can accurately simulate the flow within the boundary layer. The completed grid division is illustrated in Fig. 5.

Prior to carrying out CFD calculations, we performed a grid independence verification to ensure the reliability of the calculation results. Fig. 6 shows the characteristic curves of C_p and Z_w obtained under single layer grid numbers of 6, 15.5, 18.8, and 24.4 thousand (corresponding to Grid1, Grid2, Grid3, Grid4 respectively). When the number of single-layer grids reaches the level of Grid3, the C_p and Z_w basically no longer change as the grid number increases. Therefore, the calculations in this paper all employ a single layer of 18.8 thousand grids to ensure the calculation results are independent of the grid number.

The ANSYS CFX 19.0 is adopted for solving the Reynolds Averaged Navier-Stokes (RANS) equations. Previous studies [61–63] have shown that the Shear Stress Transport (SST) turbulence model can accurately calculate the aerodynamic performance of turbine cascades under both design and off-design conditions. Therefore, this paper adopts the SST model to simulate the pressure distribution of turbine blades. The working medium is set as ideal gas (with a gas-oil ratio of 0.019). The total temperature (1066 K), total pressure (205.1 kPa), and flow incidence angle are specified at the inlet of the computational domain, while the static pressure (135 kPa) is applied at the outlet. The upper and lower walls of the blades are designated as translational boundaries, and the blade surfaces are set as adiabatic walls with no slip. Under the

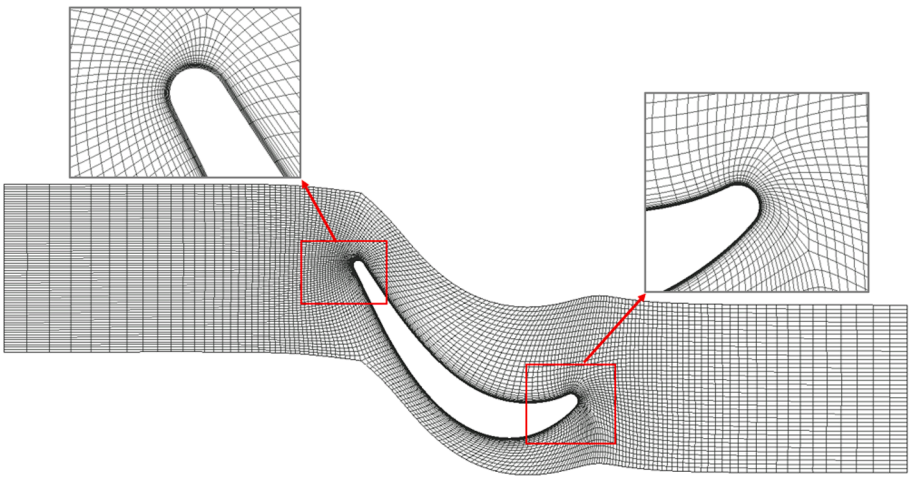
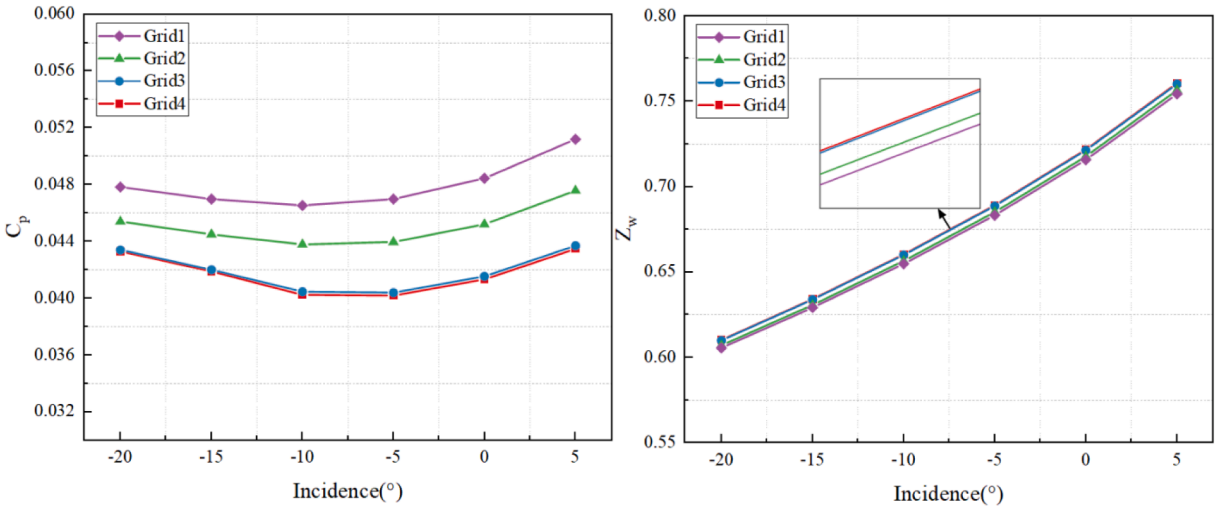


Fig. 5. Mesh local views of the blade profile.



(a) Total pressure loss coefficient (b) Zweifel coefficient

Fig. 6. Characteristic curves of total pressure loss coefficient and Zweifel coefficient under different grid levels and incidence angles.

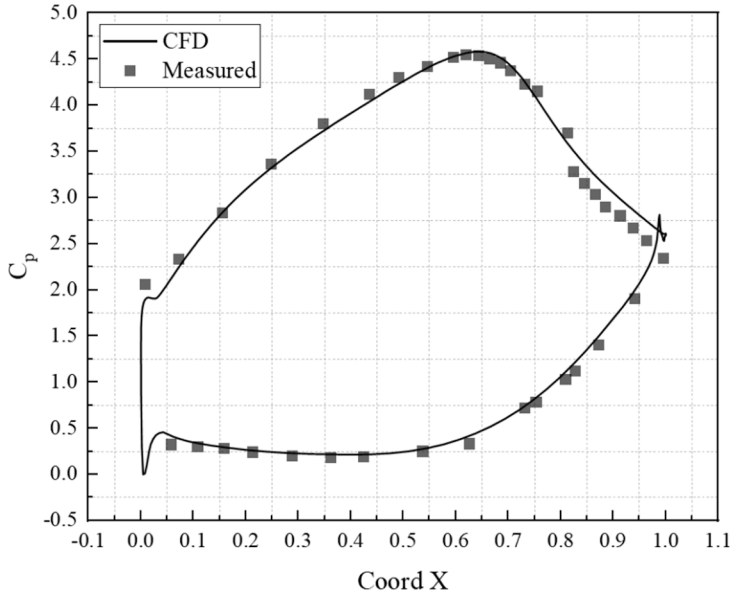


Fig. 7. Comparison between experimental measurement data [64] and CFD simulation result.

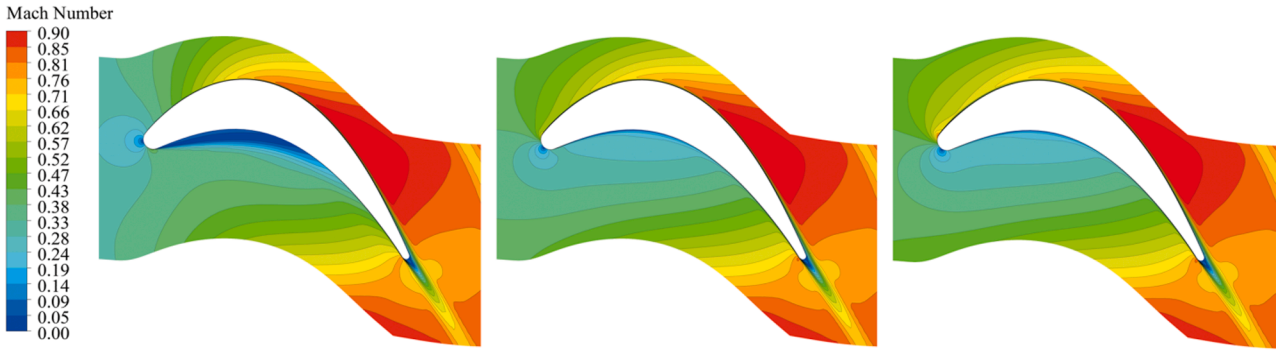


Fig. 8. Mach number distributions within the cascade at different incidence angles: −20° (left), 0° (center), and +5° (right).

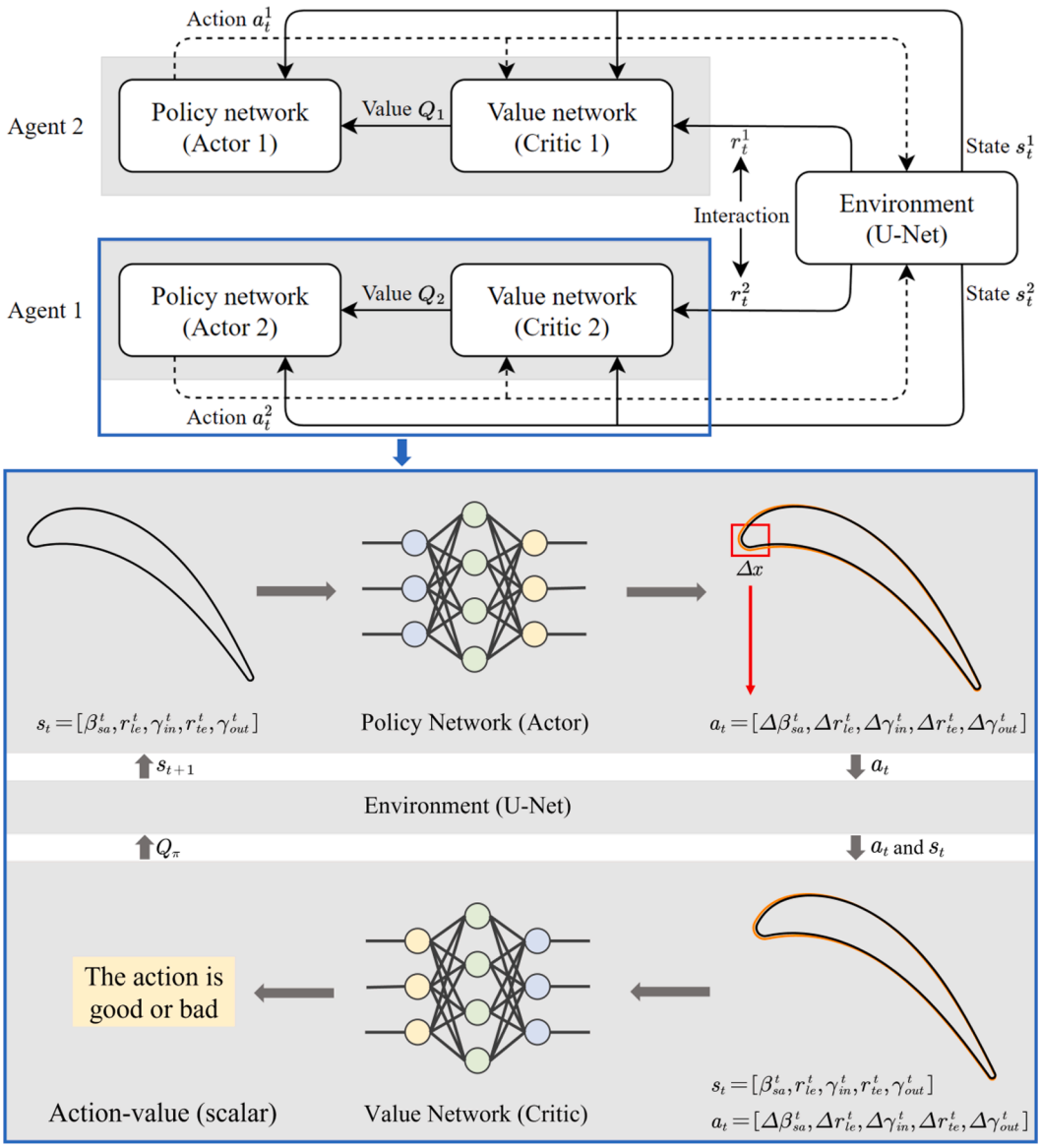


Fig. 9. The MARL process based on the Actor-Critic architecture.

aforementioned CFD settings, Fig. 7 displays a comparison between the experimental measurement results and the CFD calculation results. The comparison indicates that the grid division strategy and CFD settings in this study are reasonable, demonstrating a high consistency with the experimental measurement data.

It is worth noting that the dataset used in this study has a total of five different working conditions, which differ in the incidence angle (α) of the airflow. The range of α variation is $-20°$, $-15°$, $10°$, $0°$, and $+5°$. This range covers the majority of operating conditions encountered in actual gas turbine operations. Within this range of incidence angles, the Mach number distributions within the cascade are shown in Fig. 8. As seen from the figure, except for some flow separation on the pressure side at extreme negative incidence angles, there is almost no apparent flow separation under other conditions. This ensures the reliability of the CFD calculation results. In order to generate a dataset with good uniformity, this study employs Latin Hypercube Sampling (LHS) for data generation [65]. In previous studies [66,67], LHS has been proven to be an efficient sampling method. Compared to other sampling methods (such as random sampling or systematic sampling), LHS typically covers the entire parameter space with a smaller sample size. Under this sampling strategy, a total of 2500 sets of samples were generated, with 85 % (2125 sets of samples) used for network training, and 15 % (375 sets of samples) employed for network performance testing.

After establishing the dataset, the Pytorch open-source library in Python was utilized to train a U-Net network [68]. The trained U-Net will interact with the environment and agents of MARL. Taking agent 1 as an example, their interaction process is as follows:

$$s_t = \left[\beta_{sa}^t, r_{le}^t, \gamma_{in}^t, r_{te}^t, \gamma_{out}^t\right] \tag{19}$$

s_t denotes the state of the environment at time step t. The action is defined as:

$$a_t = \left[\Delta\beta_{sa}^t, \Delta r_{le}^t, \Delta\gamma_{in}^t, \Delta r_{te}^t, \Delta\gamma_{out}^t\right] \tag{20}$$

$\Delta\beta_{sa}^t, \Delta r_{le}^t, \Delta\gamma_{in}^t, \Delta r_{te}^t, \Delta\gamma_{out}^t$ represent the change variables of these five blade parameters. After executing a_t, the environment takes s_t as the input of U-Net to obtain the performance evaluation indicators (C_p and Z_w) of the blade, which are fed back to the agents as rewards. Then, the environment returns the next state s_{t+1}. This process continues until the maximum time step t_{max} is reached.

$$s_{t+1} = \left[\beta_{sa}^t + \Delta\beta_{sa}^t, r_{le}^t + \Delta r_{le}^t, \gamma_{in}^t + \Delta\gamma_{in}^t, r_{te}^t + \Delta r_{te}^t, \gamma_{out}^t + \Delta\gamma_{out}^t\right] \tag{21}$$

Table 2
Detailed structures of the value and policy networks.

Layer of policy network	Number of neurons	Layer value network	Number of neurons
Input layer	5	Input layer	10
Hidden layer 1	128	Hidden layer 1	128
Hidden layer 2	128	Hidden layer 2	128
Hidden layer 3	128	Hidden layer 3	128
Hidden layer 4	128	Hidden layer 4	128
Output layer	5	Output layer	1

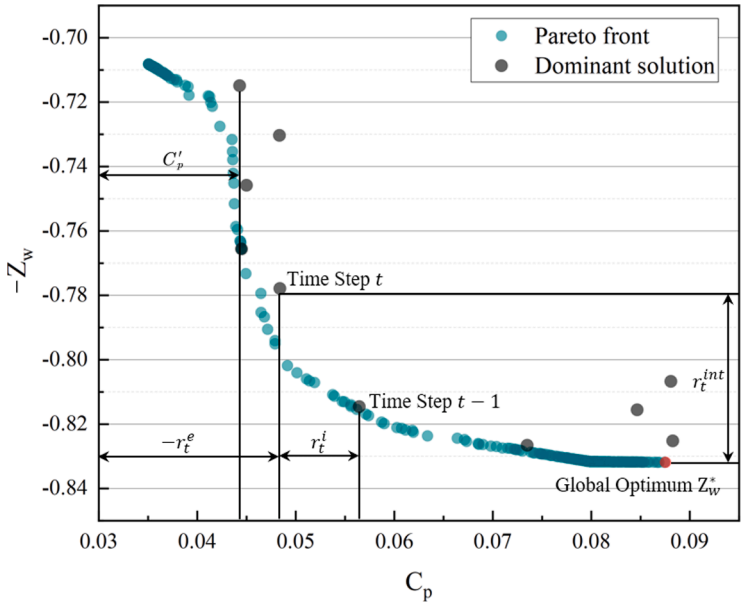


Fig. 10. Illustration of the computation of different types of rewards in the CR.

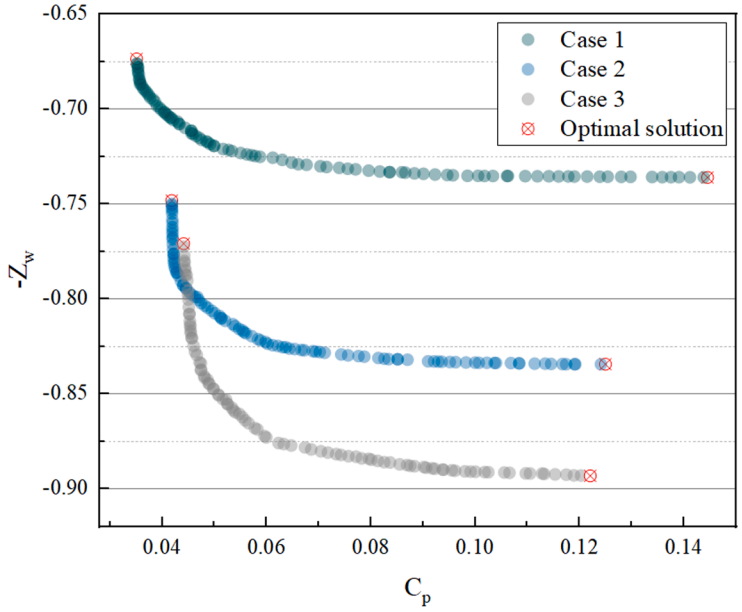


Fig. 11. Comparison of global optimal solutions under different constraints and working conditions.

Table 3
Detailed structures of the ANN network.

Layer of ANN network	Number of neurons
Input layer	x
Hidden layer 1	16
Hidden layer 2	16
Hidden layer 3	16
Output layer	2

3.3. Learning strategy

Learning strategy is the core of RL, as it defines how agents can continuously learn and improve themselves through the interaction process with environment. This paper applies the multi-agent proximal policy optimization (MAPPO) algorithm to optimize the policy function of multiple agents. MAPPO algorithm is a policy optimization algorithm suitable for multi-agent developed based on PPO algorithm. The PPO algorithm was proposed by OpenAI in 2017 [69]. It performs well in solving problems of continuous action space and high-dimensional state space in RL. Its main feature is that by limiting the step size of each update while ensuring the effectiveness of samples, it enhances the stability of policy updates. Due to its superior performance and stability, the PPO algorithm has been extensively applied in various RL applications [70–72].

The PPO used in this study is a deep RL algorithm based on the Actor-Critic (AC) architecture, whose structure as shown in Fig. 9. PPO-AC consists of two parts: the policy network and the value network. The value network represents the Actor in PPO-AC, while the policy network represents the Critic. The PPO algorithm based on the AC architecture includes both policy learning and value learning. Its parameter update is mainly based on the state value function. After introducing the neural network, the state value function can be written as:

$$V_\pi(s) = \sum_a \pi(a|s;\theta)\cdot Q_\pi(s,a;w) \tag{22}$$

In the above formula, θ represents the parameters of the policy network, while w represents the parameters of the value network. The policy network is primarily responsible for learning the optimal policy function π^*, with supervision provided by the value network Q_π. A larger value of Q_π indicates that taking action a_t based on the current state s_t is more advantageous. The value network Q_π is responsible for scoring the current action a_t, with supervision provided by the environment. As the parameters are updated, the scoring of Q_π becomes increasingly accurate. The pseudocode of the PPO-AC is presented in Algorithm 1.

Algorithm 1: PPO-AC.

1 initialization actor network π_θ and critic network Q_w with random parameters θ, w;
2 initialization state s_t, maximum number of iterations n_{max}, iteration count $n = 0$;
3 initialization maximum time step $t_{max} = 10$, time step count $t = 0$, replay buffer $\mathcal{B}$;
4 initialization actor and critic network learning rate α, β, discount factor γ;
5 **while** $n < n_{max}$ **do**
6 $n \leftarrow n + 1$;
7 **repeat**
8 $t \leftarrow t + 1$;
9 obtain a_t based on s_t and actor network $a_t \sim \pi(a \mid s;\theta)$;
10 agent performs a_t, the environment gives a new state s_{t+1} and reward r_t ;
11 based on s_{t+1}, a new action is computed $a_{t+1} \sim \pi(a \mid s_{t+1};\theta)$;
12 critic network scores the action $q_t = Q(s_t, a_t; w_t)$, $q_{t+1} = Q(s_{t+1}, a_{t+1}; w_t)$;
13 compute the TD error $\delta_t = q_t - (r_t + \gamma \cdot q_{t+1})$;
14 update the critic network using the TD algorithm $w \leftarrow w - \alpha \cdot \delta_t \cdot \frac{\partial q(s_t,a_t;w)}{\partial w}$;
15 update the actor network using the PPO algorithm $\theta_{t+1} \leftarrow \theta_t + \beta \frac{\partial \log \pi(a_t|s_t,\theta)}{\partial \theta}$;
16 store data of (s_t, a_t, r_t)in $\mathcal{B}$;
17 **until** $t > t_{max}$;
18 **end**

The replay buffer in the line 3 of the algorithm is a common technique in RL [41]. It is a technique for storing and replaying experience data. The replay buffer helps to decouple the time correlation of the data, make full use of historical experience, reduce the waste of training

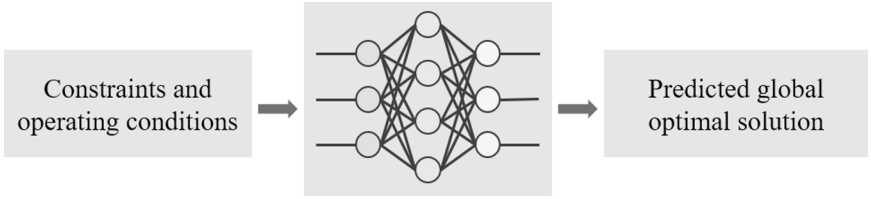


Fig. 12. The working process of ANN in the reward function.

samples, and balance the data distribution. The detailed update process of the policy network and value network in line 14 and 15 of the algorithm using the TD algorithm and PPO algorithm refer to References [73] and [69].

As the parameters of the value and policy networks are updated, the policy function will always return the action a_t that maximizes the value of V_π when given the current state s_t. As stated in Section 2.1, a larger value of V_π indicates a higher expected cumulative reward for the agent, which means that the current policy is better.

It is worth mentioning that the MAPPO algorithm used in this paper differs slightly from the original PPO algorithm. The implementation details can be found in Reference [74]. In our study, there are a total of two agents, each with a set of networks (which includes a policy network and a value network). The detailed structures of these networks are presented in Table 2. The two agents do not share network parameters during the training process, and their interaction is achieved through the reward function, as detailed in the following section.

3.4. Reward shaping

Rewards play a crucial role in driving the learning process of agents. A well-designed reward can help the agent quickly discover the optimal policy. To apply MARL to DMOPs, this paper proposes two types of reward functions: a comprehensive reward (CR) and an embedded comprehensive reward based on artificial neural network (CR-ANN).

(1) Comprehensive reward

The definition of CR in this study is as follows:

$$r_t = r_t^e + \kappa_1 r_t^i + \kappa_2 r_t^{int} \tag{23}$$

$$\text{agent 1}\begin{cases} r_t^e = -C_p^t \\ r_t^i = C_p^{t-1} - C_p^t \\ r_t^{int} = \dfrac{-1}{Z_w^t - Z_w^*} \end{cases} \tag{24}$$

$$\text{agent2}\begin{cases} r_t^e = Z_w^t \\ r_t^i = Z_w^t - Z_w^{t-1} \\ r_t^{int} = \dfrac{-1}{C_p^* - C_p^t} \end{cases} \tag{25}$$

In the above equation, r_t^e represents the extrinsic reward of the agent, which is the direct reward obtained by the agent at time t. Specifically, for agent 1, $r_t^e = -C_p$, and for agent 2, $r_t^e = Z_w$. r_t^i represents the intrinsic reward obtained by the agent, which is equal to the difference between the external rewards of the agent at time t and time $t-1$. r_t^{int} represents the interactive reward, which encourages the agents to collaborate with each other to find the optimal solution. κ_1 and κ_2 are weight coefficients. To explain the above three types of rewards in detail, this paper uses agent 1 as an example to illustrate the process of solving these rewards, as shown in Fig. 10 and Algorithm 2.

Algorithm 2: Reward function for agent1

initialization maximum number of iterations n_{max}, iteration count $n = 0$;
initialization episode maximum time step t_{max}, time step count $t = 0$;
initialization set local optimum $C_p' = 0$, global optimum $C_p^* = 0$;
initialization set $r_0^e = C_p^0$, $r_0^i = 0$, randomly sample a_0 from action space;
while $n < n_{max}$ **do**
  while $t < t_{max}$ **do**
    agent1 performs a_t, the environment gives s_{t+1}, C_p^t and Z_w^t;
    agent2 gives the global optimum Z_w^*;
    set reward $r_t^e = C_p^t$, $r_t^{int} = 0$;
    if $t > 0$ **then**
      $r_t^i = C_p^t - C_p^{t-1}$
    end
    if $C_p^t > C_p'$ **then**
      $r_t^{int} = \frac{-1}{Z_w^t - Z_w^*}$;
      $C_p' = C_p^t$;
    end
    if $C_p^t > C_p^*$ **then**
      $r_t^i = 10 r_{in}^t$;
      $C_p^* = C_p^t$;
    end
    $r_t = r_t^e + \kappa_1 r_t^i + \kappa_2 r_t^{int}$;
    the actor network gives a_{t+1} based on r_t;
    $t \leftarrow t + 1$;
  end
  $n \leftarrow n + 1$;
end

It is worth noting that the goal of RL is to maximize the cumulative reward, while in the optimization process, we aim for a smaller value of C_p. To solve this contradiction, we normalize C_p to the interval $[-1, 0]$ and Z_w to the interval $[0, 1]$. In Algorithm 2, both C_p and Z_w are rewards that have been normalized. This results in a discrepancy between r_t^i in Algorithm 2 and the expression in Eq. (24).

In Algorithm 2, $C_p^{'}$ represents the best C_p found in each episode, and C_p^* represents the best C_p found in all episodes. From the reward setting, we can see that the intrinsic reward r_t^i is designed to motivate the agent to spontaneously seek the optimal solution. If the agent finds a better solution C_p at time step $t+1$ compared to the one found at time step t, the agent receives a positive reward. Otherwise, it receives a negative reward. $r_{c_p z_w}^t$ is to motivate the agent to spontaneously search for non-dominated solutions. When solution C_p^t is better than $C_p^{'}$ at time t, we consider that the solution has the potential to become a non-dominated solution. We measure the distance between Z_w^t and the global optimum Z_w^* (which is obtained by agent 2) to determine the reward r_t^{int} that the agent receives. The smaller the distance, the greater r_t^{int} obtained by the agent. Lines 17–20 of the algorithm encourage the agent to explore better global optimal solutions. When C_p^t surpasses the historical global optimum C_p^*, we increase the intrinsic reward by a factor of 10. This prevents the agent from getting trapped in local optimal solutions.

The reward function of agent 2 is consistent with the above settings, except that C_p in Algorithm 2 is replaced by Z_w, and Z_w is changed to C_p. The reward function design indicates that agent 1 is responsible for finding the optimal value of C_p, while agent 2 is responsible for finding the optimal value of Z_w. Their interaction is achieved through the interaction reward r_t^{int}.

(2) Embedded comprehensive reward based on artificial neural network

When solving DMOPs, as working conditions and geometric constraints change, the globally optimal solution defined in Algorithm 2 may become invalid. Because under each working condition and constraint, there corresponds an optimal C_p^* and Z_w^* (as shown in Fig. 11), while the C_p^* and Z_w^* found in Algorithm 2 are the globally optimal solutions optimal under all working conditions and geometric constraints. Therefore, in the later stages of optimization, lines 14 and 17–20 of Algorithm 2 may fail, leading to a risk of the algorithm getting trapped in optimal solutions. To alleviate this problem, we propose an embedded comprehensive reward algorithm based on ANN, as shown in Algorithm 3.

Algorithm 3: Reward function for agent1

1 initialization maximum number of iterations n_{max}, iteration count $n = 0$;
2 initialization episode maximum time step t_{max}, time step count $t = 0$;
3 initialization set local optimum $C_p' = 0$, global optimum $C_p^* = 0$, $Z_w^* = 0$;
4 initialization set $r_0^e = C_p^0$, $r_0^i = 0$, randomly sample a_0 from action space;
5 initialization ANN f_ξ with random parameters ξ;
6 **while** $n < n_{max}$ **do**
7 **while** $t < t_{max}$ **do**
8 agent1 performs a_t, the environment gives s_{t+1}, C_p^t and Z_w^t;
9 ANN predict the best C_p and Z_w, $C_p^{best}, Z_w^{best} = f(s_t, \xi)$;
10 set reward $r_t^e = C_p^t$;
11 set reward $r_t^i = C_p^t - C_p^{best}$;
12 **if** $Z_w^t < Z_w^*$ **then**
13 $r_t^{int} = Z_w^t - Z_w^{best}$;
14 **end**
15 **if** $C_p^t > C_p^*$ **then**
16 $C_p^* = C_p^t$;
17 $Z_w^* = Z_w^t$;
18 **end**
19 **if** $C_p^t > C_p'$ **then**
20 $C_p' = C_p^t$;
21 **end**
22 $r_t = r_t^e + \kappa_1 r_t^i + \kappa_2 r_t^{int}$;
23 the actor network gives a_{t+1} based on r_t;
24 $t \leftarrow t + 1$;
25 **end**
26 update the ANN based on the C_p' from agent1 and Z_w' from agent2;
27 $n \leftarrow n + 1$;
28 **end**

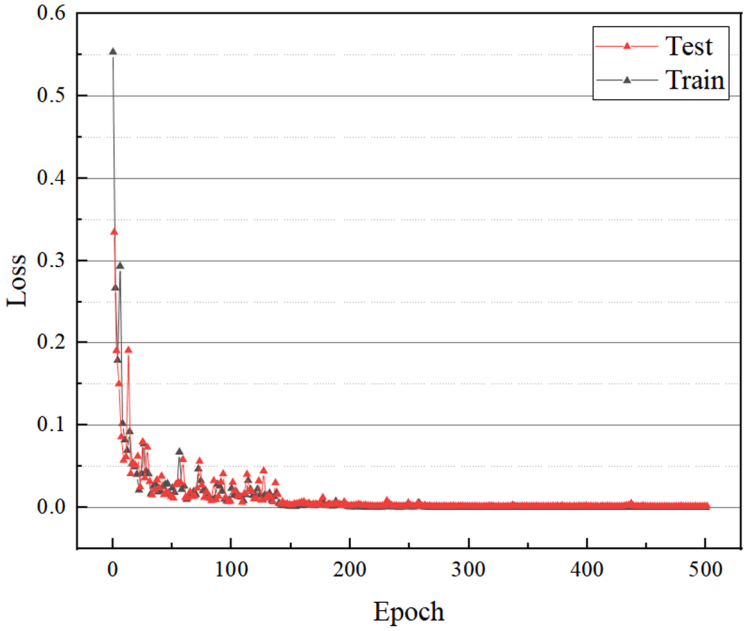
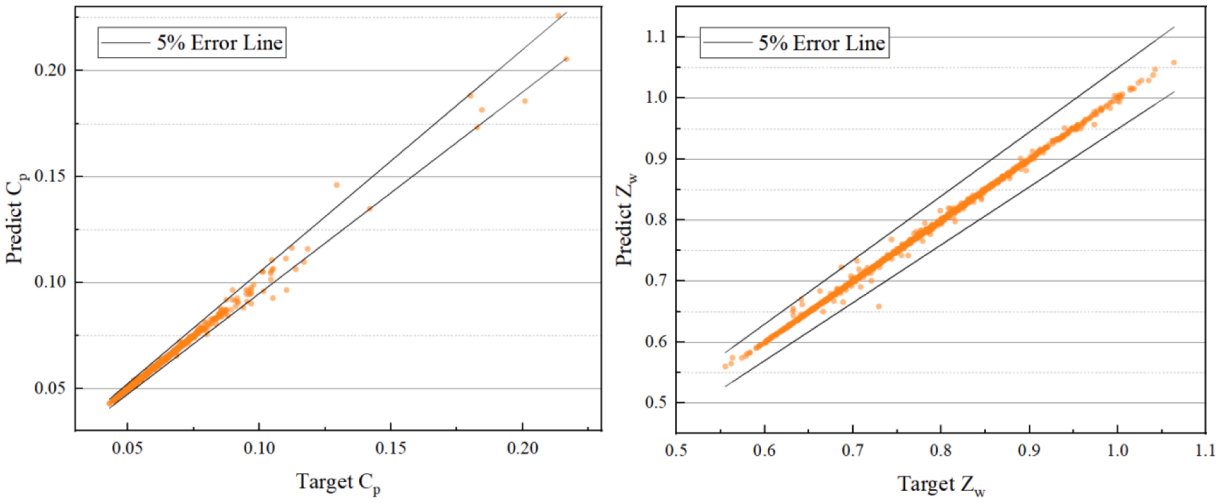


Fig. 13. The convergence curve of the loss function for the U-Net.

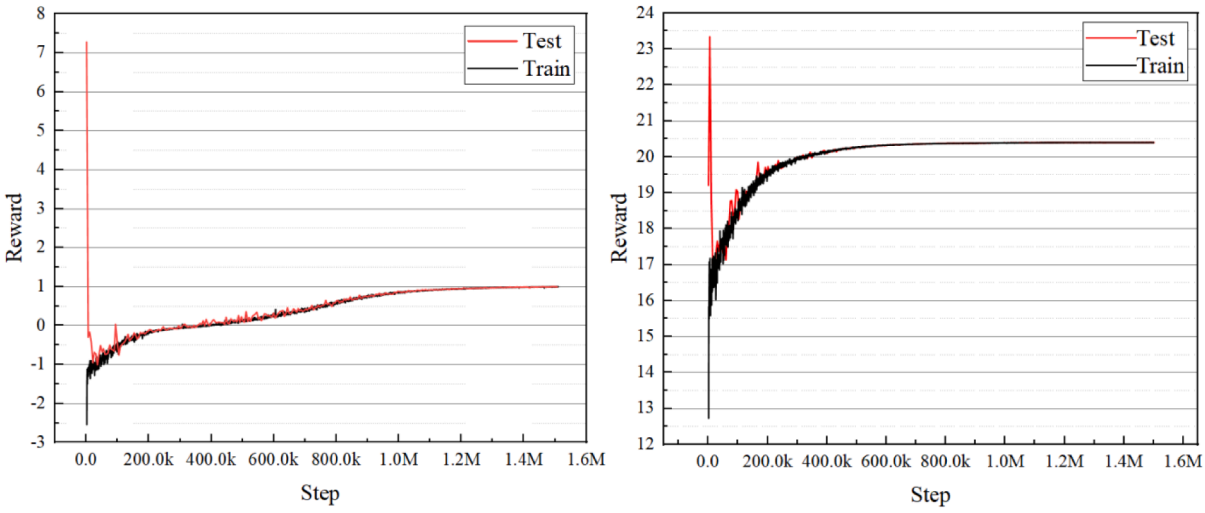


(a) The distribution of prediction errors for C_p (b) The distribution of prediction errors for Z_w

Fig. 14. Absolute error distribution of U-Net on the test set.

The detailed structure of the ANN network in Algorithm 3 is presented in Table 3. The number of input variables in the ANN network is determined by the number of geometric constraints and changes in blade working conditions. For example, when only one geometrical constraint and the inlet airflow angle of the blade change, $x = 2$. The working process of the ANN is presented in Fig. 12, where it predicts the optimal C_p (C_p^{best}) and Z_w (Z_w^{best}) according to the current state s_t. In the early stage of training, ANN's prediction of the C_p^{best} and Z_w^{best} is not accurate, but as the training progresses, ANN's prediction will get better and better. Compared with Algorithm 2, Algorithm 3 changes the settings of intrinsic reward and interaction reward. When we know the optimal values C_p^{best} and Z_w^{best} corresponding to each working condition, we can directly measure the difference between the current solution and the optimal solution as an intrinsic reward. This intrinsic reward setting makes the training process more stable. Similarly, the interactive reward is defined using the difference between the predicted and actual values, which replaces the reciprocal used in Algorithm 2 to avoid the oscillation of reward during training.

As analyzed above, Algorithm 3 accomplishes the communication between agent 1 and agent 2 through an ANN network. Specifically, we consider the local optimal solutions C_p^i and Z_w^i obtained by both agents in each episode as the optimal solutions for the given operating condition, and continuously update the parameters of the ANN. Eventually, the ANN provides valuable guidance for the training process of both agents based on different operating conditions and constraints. The comparison of the optimization results obtained using these two reward functions can be found in Section 4.



(a) The reward convergence process of agent 1 (b) The reward convergence process of agent 2

Fig. 15. The convergence curve of rewards during the training process.

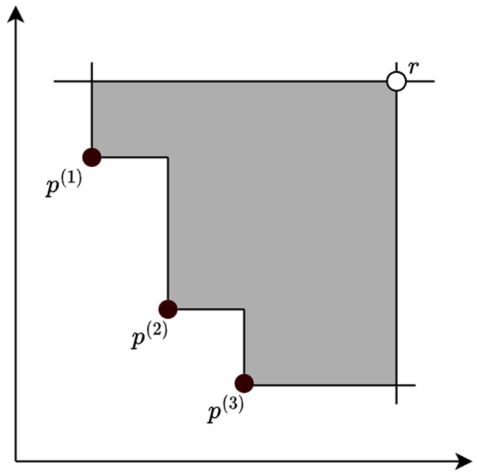


Fig. 16. The hypervolume indicator in the two-objective case.

4. Experiments and result analysis

4.1. Validation of prediction accuracy of U-Net

Fig. 13 illustrates the convergence of the loss function during the training of the U-Net network. Here, the loss is defined as the root mean square error (RMSE) between the predicted and actual values, as shown in Eq. (26).

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{\left(\Omega_{obs,i} - \Omega_{model,i}\right)^2}{n}} \tag{26}$$

In the above equation, Ω_{obs} is the C_p and Z_w obtained from CFD calculations, and Ω_{model} denotes the predicted C_p and Z_w from the U-Net. As depicted in Fig. 13, throughout the training process, the error of the U-Net model gradually converges to 0.0011 for the training set and 0.0018 for the test set. The convergence results indicate that the network does not experience overfitting on the small sample dataset used in this paper, which confirms the remarkable generalization ability of the U-Net structure. For further quantitative analysis of its prediction results, Fig. 14 shows the distribution of absolute errors predicted by the U-Net network on the testing set. The error distribution indicates that the U-Net's predictions for C_p and Z_w are generally within a 5 % error band.

The high-precision prediction of blade performance by the U-Net ensures the correctness of the optimization algorithm's search direction. After completing the model training, we constructed a MARL environment based on the Python gym open-source library [75].

4.2. Multi-objective optimization based on MARL

When using Algorithm 2 as the reward function with $\kappa_1 = 10$ and $\kappa_2 = 0$, the reward changes on the training and testing sets during the training process are shown in Fig. 15. After training for 150 epochs (with 10,000 steps of interaction with the environment per epoch), the training set rewards of agent 1 and agent 2 converge to 1.016, and their testing set rewards converge to 20.401.

After the training is completed, to comprehensively assess the optimization performance of the algorithm, this study employs the Pareto hypervolume as a metric to measure the extent to which the solution set approaches the true Pareto front in the objective space and the uniformity of the solution set distribution [76]. As shown in Fig. 16, the Pareto hypervolume calculates the area enclosed between the Pareto front and the reference point. The larger the Pareto hypervolume, the better the uniformity of the Pareto front distribution and the higher the solution quality.

In this paper, the benchmark for comparing the optimization performance of the MARL is the NSGA-II algorithm [77]. NSGA-II is one of the most widely used algorithms in multi-objective optimization, possessing both robust global search capabilities and remarkable solution speeds [78]. Fig. 17 shows a comparison of the Pareto front found by MARL and NSGA-II. The Pareto hypervolume of MARL is 0.7523, and the Pareto hypervolume of NSGA-II algorithm is 0.7600. Although the Pareto volume found by RL is slightly smaller than that of NSGA-II, its search for the optimal C_p value is better than NSGA-II.

When we consider the interactive reward in the comprehensive reward function with $\kappa_1 = 10$ and $\kappa_2 = 10^{-4}$, the Pareto fronts found by MARL and NSGA-II are shown in Fig. 17. After adding interactive reward, the Pareto hypervolume found by RL is 0.7670, which improves the optimization performance by about 2 %. Moreover, the optimal value of C_p found by RL is also better than that of NSGA-II. This demonstrates that the interactive reward is helpful for the optimization process of MARL. Fig. 18 shows the changes in the external reward of agent 1 when the intrinsic reward is added or removed during the

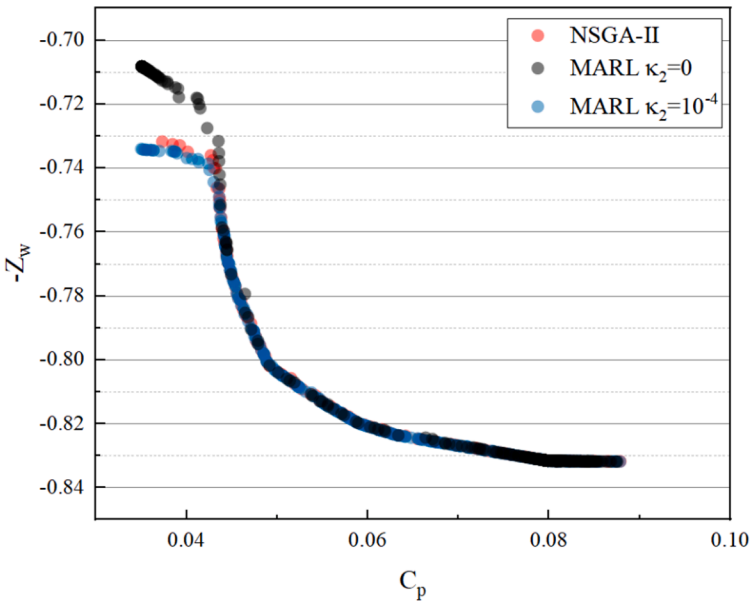


Fig. 17. Comparison of optimization results between MARL and NSGA-II.

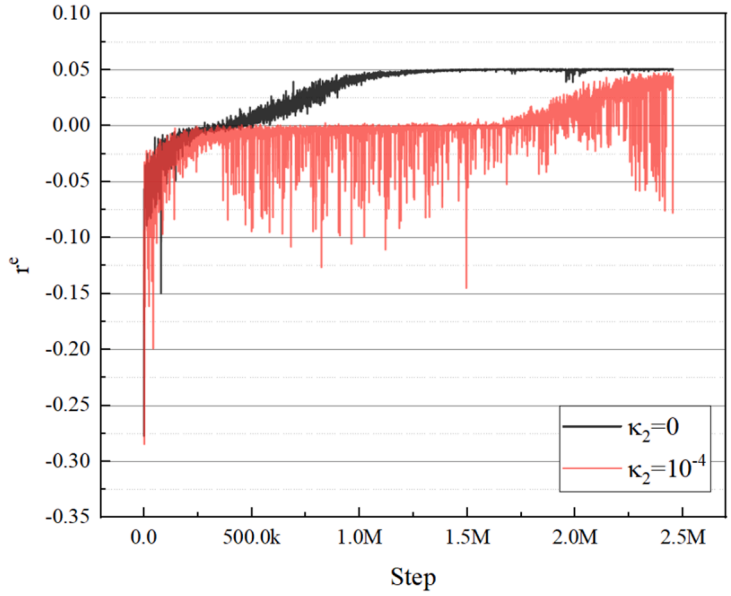


Fig. 18. Convergence curve of the r^e for agent 1 under different reward settings.

training process. After adding interactive reward, the convergence speed of the training process slowed down (converging around the 240th epoch), and the external reward fluctuated greatly in the later stage of training. Since the reward structure has changed, their final convergence results are different. After adding the interactive reward, the extrinsic reward converged to 0.0462, while without the interactive reward it converged to 0.0495.

The changes in the external reward of agent 2 are similar to those of agent 1 after introducing interactive reward. The convergence rate of the training process slowed down, and the peak of the external reward convergence decreased. Additionally, significant fluctuations are observed in the later stages of training. This may be due to the additional constraints that interactive rewards impose on the agents. When the solution sought by the agents is dominated, they will not receive rewards, which makes them more focused on finding non-dominated solutions. This results in a smaller final convergence value of the external reward. In the later stages of training, most non-dominated solutions have been found, and the intrinsic reward plays a dominant role in motivating the agents to explore a more optimal Pareto front. This corresponds to the change process of the Pareto front from the 230th to the 240th epoch when $\kappa_2 = 10^{-4}$ in Fig. 19.

4.3. Dynamic multi-objective optimization based on MARL

The analysis in the previous section was carried out under a single working condition and without constraints. However, in the initial design stage of turbine blades, due to the limitation of structural strength, it is often necessary to constrain some geometric parameters. Since different geometric constraints correspond to different optimization problems, this is a dynamic optimization problem. In order to apply

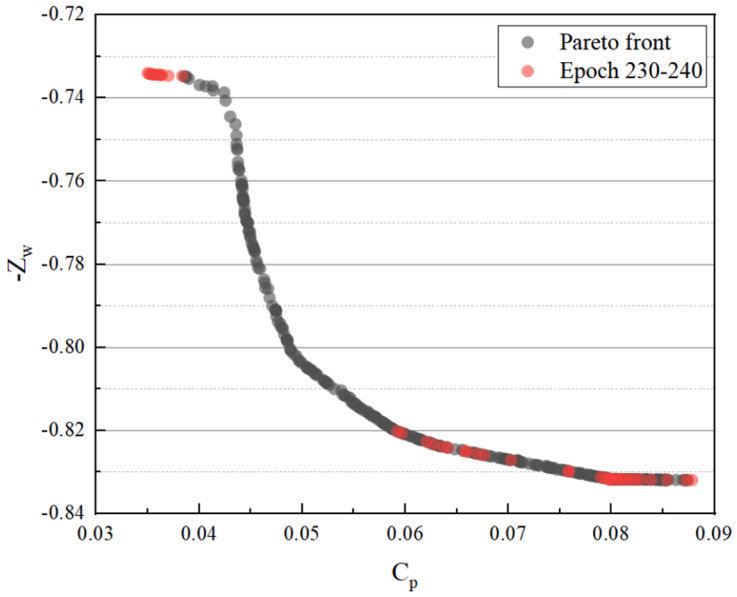
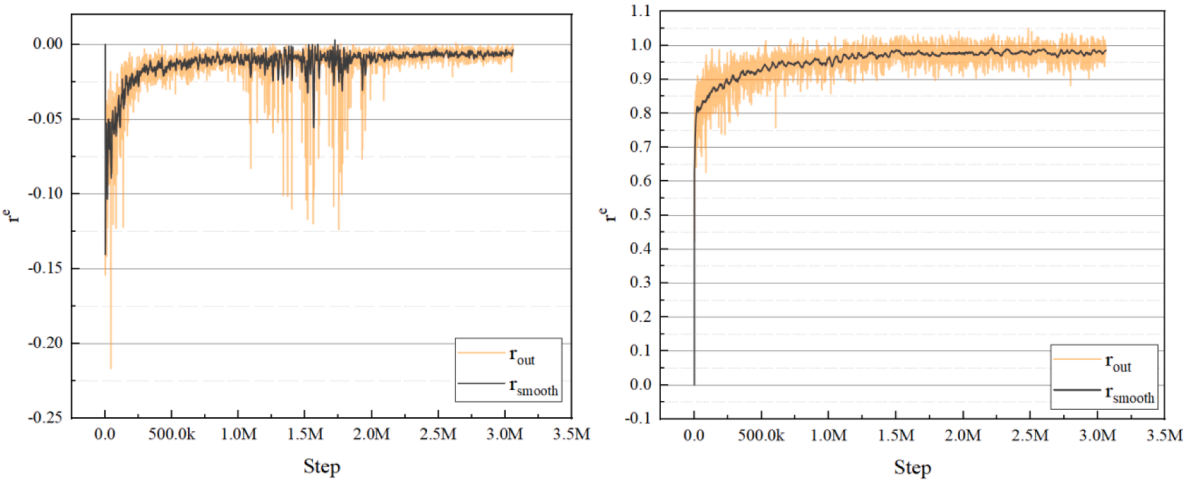


Fig. 19. Changes in the Pareto front during the training process.



(a) The reward convergence process of agent 1 (b) The reward convergence process of agent 2

Fig. 20. Reward convergence curves during training.

MARL to the DMOPs of turbine blade, this paper takes the leading edge radius as an example to illustrate how to optimize turbine blade profiles under certain geometric constraints.

Different from the MARL environment defined in Section 3.2, when considering the leading edge radius constraint, the leading edge radius of the blade is constant in each episode. At this time, the environment state at time t is defined as:

$$s_t = \left[\beta_{sa}^t, \gamma_{in}^t, r_{te}^t, \gamma_{out}^t\right] \tag{27}$$

The definition of environmental action is as follows:

$$a_t = \left[\Delta\beta_{sa}^t, \Delta\gamma_{in}^t, \Delta r_{te}^t, \Delta\gamma_{out}^t\right] \tag{28}$$

The state at time $t+1$ is as follows:

$$s_{t+1} = \left[\beta_{sa}^t + \Delta\beta_{sa}^t, \gamma_{in}^t + \Delta\gamma_{in}^t, r_{te}^t + \Delta r_{te}^t, \gamma_{out}^t + \Delta\gamma_{out}^t\right] \tag{29}$$

After completing an episode, the leading edge radius of the next episode is randomly sampled. The above configuration ensures that the leading edge radius is fixed within a given episode, but varies across different episodes. This arrangement allows the agent to adapt to optimization problems with different constraints.

According to the results of Section 4.2, when $\kappa_1 = 10$ and $\kappa_2 = 10^{-4}$, the external reward convergence curves of agent 1 and agent 2 are shown in Fig. 20. After 300 epochs of training, the extrinsic rewards converged to −0.0036 for agent 1 and 0.985 for agent 2. Fig. 21 and Table 4 show the comparison of the Pareto front found by MARL and NSGA-II after the training is completed, when the current radius is 0.4 and 1.0. From the comparison, it can be seen that in DMOPs, the MARL based on CR performs worse than NSGA-II, but its optimization time is greatly reduced. On average, MARL takes 0.12 s per optimization problem, which is approximately 1/57 of the time required by the NSGA-II. As analyzed in Section 3.2, the reason why the optimization performance of MARL is inferior to that of NSGA-II is that with changes in geometric constraints, the composite reward can no longer effectively guide the intelligent agent to find non-dominant solutions, leading to MARL being trapped in a local optimum.

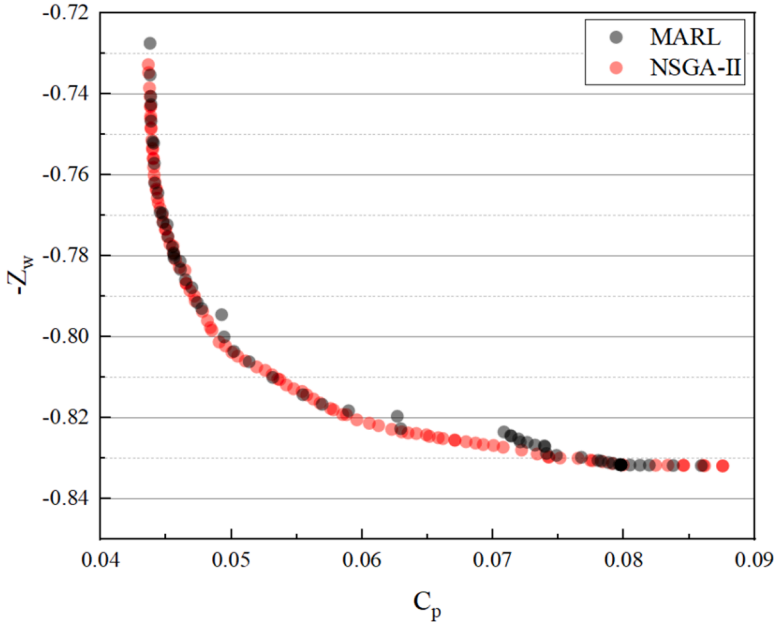


Fig. 21. The Pareto front found by MARL-CR and NAGA-II when the leading edge radius is 0.4.

Table 4
Performance comparison between MARL-CR and NSGA-II under different leading edge radius.

r_{le}	NSGAII	MARL	T_{NSGAII}(s)	T_{MARL}(s)
0.40	0.8611	0.8465	6.92	0.12
0.55	0.7613	0.7376	6.75	0.11
0.70	0.7737	0.7455	6.81	0.12
0.85	0.7758	0.7483	6.71	0.12
1.00	0.8106	0.8012	6.70	0.13
Average	0.7965	0.7758	6.78	0.12

Table 5
Performance comparison between MARL-CRANN and NSGA-II under different leading edge radius.

r_{le}	NSGAII	MARL	T_{NSGAII}(s)	T_{MARL}(s)
0.40	0.8611	0.8649	6.92	0.13
0.55	0.7613	0.7611	6.75	0.10
0.70	0.7737	0.7743	6.81	0.11
0.85	0.7758	0.7764	6.71	0.12
1.00	0.8106	0.8105	6.70	0.12
Average	0.7965	0.7974	6.78	0.12

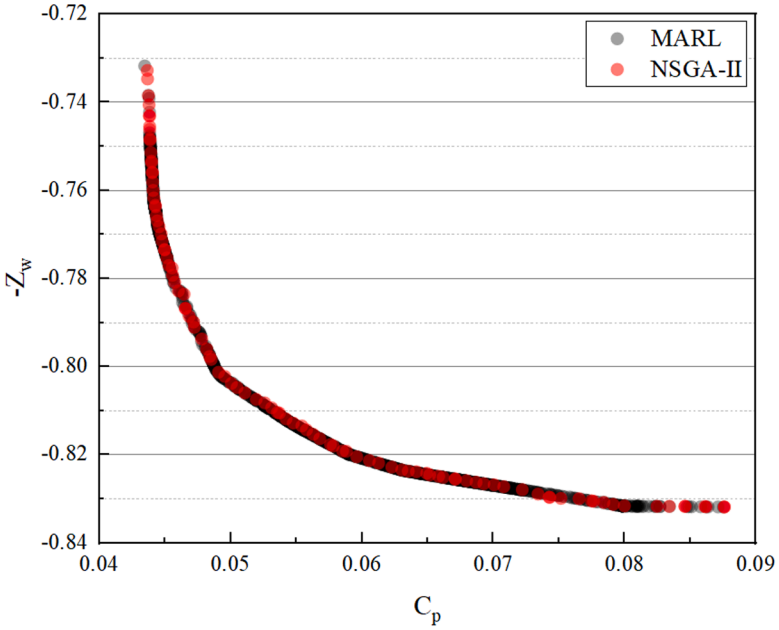


Fig. 22. The Pareto front found by MARL-CRANN and NAGA-II when the leading edge radius is 0.4.

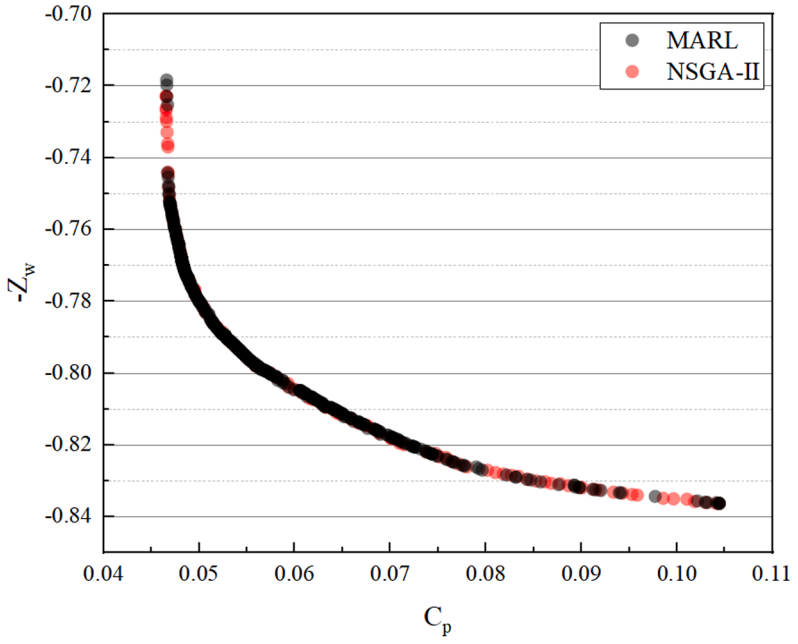


Fig. 23. Comparison of the Pareto front found by NSGA-II and MAOP when $r_{le} = 1$ and $\alpha = 5^{\circ}$.

Table 6
Performance comparison between NSGA-II and MARL under different geometric constraints and working conditions.

r_{le}	$\alpha = -20^{\circ}$		$\alpha = -10^{\circ}$		$\alpha = 0^{\circ}$		$\alpha = 5^{\circ}$	
	NSGAII	MARL	NSGAII	MARL	NSGAII	MARL	NSGAII	MARL
0.40	0.5631	0.5672	0.8825	0.8839	0.9023	0.9021	0.8807	0.8807
0.55	0.5878	0.5880	0.8370	0.8382	0.8828	0.8837	0.8548	0.8562
0.70	0.3957	0.3943	0.8316	0.8321	0.8791	0.8793	0.8558	0.8575
0.85	0.6723	0.6723	0.8226	0.8193	0.8417	0.8418	0.8587	0.8590
1.00	0.5560	0.5479	0.8441	0.8449	0.8267	0.8270	0.8241	0.8224
Average	0.5550	0.5540	0.8436	0.8437	0.8665	0.8668	0.8548	0.8548

Table 7
Time consumption of MARL in training and inference stages.

Model	Train (s)	Test (s)
MARL-CR	249.0	0.12
MARL-CRANN	501.6	0.12

To overcome the issue of the algorithm's declining optimization performance, this study uses CR-ANN to perform multi-objective optimization on blade profiles, instead of using CR. The optimization results are shown in Fig. 22 and Table 5. It is evident from the optimization results that replacing CR with CR-ANN did not result in a significant change in the optimization time of MARL. For these five optimization problems, the Pareto hypervolume average value obtained by the MARL based on CR is approximately 0.7758, while the hypervolume obtained by the MARL based on CR-ANN is 0.7974. The improved reward function enhances the multi-objective optimization performance of MARL by 2.8 %. In addition, compared to NASG-II, MARL based on CR-ANN also exhibits slightly better optimization results.

The working conditions have a significant impact on the aerodynamic performance of the blade profile. To account for the influence of operating conditions on the blade profile, this study optimizes the aerodynamic performance of the blade profile under different geometric constraints and inlet incidence angles. The definition of the state and action in the RL environment is the same as in Eq. (27) and (28). However, during training, the leading edge radius and airflow angle of the blade are unchanged within the same episode, but are randomized across different episodes.

The optimization results are shown in Fig. 23 and Table 6. Even with two dynamic variables, the MARL based on the CR-ANN can accurately find the Pareto front. Overall, the Pareto front obtained by MARL is roughly the same as that obtained by NSGA-II, with better Pareto hypervolume in most working conditions. However, MARL exhibits slightly inferior optimization results than NSGA-II at large negative attack angles. In Table 6, the average optimization time of MARL is 0.13 s, while that of NSGA-II is 6.68 s. Under similar optimization results, the optimization time cost of MARL is only 1/51 of NSGA-II. This implies that after the completion of the MARL model training, it is possible to perform real-time optimization based on different geometric constraints and blade working conditions, providing rapid and accurate guidance to designers.

In this research, the MARL models based on CR and CR-ANN were both trained on a 12 GB NVIDIA GeForce RTX 3060 GPU. Their time consumption in the training and inference stages is shown in Table 7. The data presented demonstrates that the time consumption of MARL is primarily concentrated in the training stage, but this time-consuming work can be completed ahead of time on computers. In the inference stage, the well-trained models can optimize blade shapes in real-time under arbitrary geometric constraints and inlet flow angles within a given range, without needing repeating training. This adaptive adjustment characteristic of MARL according to environmental changes is something traditional algorithms do not possess. This advantage makes it suitable for application to different turbine blade profile design and optimization problems. For new design problems, MARL can provide optimal solutions in an extremely short time, which greatly improves the efficiency of blade design.

5. Conclusion

This paper proposes a DMOP method based on MARL. Firstly, a U-Net network is used as a CFD surrogate model to evaluate the aerodynamic performance of the blade profiles. Then, it is combined with the MAPPO algorithm to perform multi-objective optimization of the turbine blade profiles. In order to improve the multi-objective optimization performance of the MAPPO algorithm, a CR strategy is designed for the multi-objective optimization problem of turbine blade profiles. This strategy promotes cooperation among agents through interactive rewards, and helps the MAPPO algorithm find a better Pareto front. To realize DMOP of blade profiles, this study designs an embedded comprehensive reward mechanism based on ANN on the basis of the CR. The policy network trained based on this reward mechanism can quickly and accurately find the Pareto front under varying inlet flow angles and geometric constraints, achieving real-time optimization. The research in this paper draws the following conclusions:

(1) The U-Net model designed in this study is capable of establishing the mapping between blade profile parameters and aerodynamic performance with a small amount of data. The relative errors in predicting C_p and Z_w are both within 5 %.
(2) Interactive reward can significantly improve the optimization performance of MARL. After adding interactive reward, the Pareto hypervolume is increased by around 2 %. For the DMOPs, CR-ANN has better optimization performance than CR, and the Pareto hypervolume increases by 2.8 %.
(3) The DMO model based on MARL can perform real-time problem solving under different geometric constraints and operating conditions after training, and its solution time cost is only about 1/51 of traditional optimization algorithm.

CRediT authorship contribution statement

Lele Li: Conceptualization, Methodology, Software, Data curation, Formal analysis, Writing – original draft, Writing – review & editing. **Weihao Zhang:** Supervision, Project administration, Funding acquisition. **Ya Li:** Supervision. **Chiju Jiang:** Conceptualization, Methodology, Validation. **Yufan Wang:** Project administration, Investigation.

Declaration of Competing Interest

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.

Data availability

Data will be made available on request.

Acknowledgments

The authors gratefully acknowledge funding support from the Program for National Natural Science Foundation of China (No. 52176033) and Science Center for Gas Turbine Project (No. P2022-B-II-009-001).

References

[1] Chen Y, Chen Y, Zhou J, Guo P, Li J. Optimization and performance study of bidirectional Savonius tidal turbine cluster with deflectors. Energ Conver Manage 2023;283:116947.
[2] Chang CCW, Ding TJ, Ping TJ, Chao KC, Bhuiyan MAS. Getting more from the wind: Recent advancements and challenges in generators development for wind turbines. Sustainable Energy Technol Assess 2022;53:102731.
[3] Aygun H, Kirmizi M, Kilic U, Turan O. Multi-objective optimization of a small turbojet engine energetic performance. Energy 2023;271:126983.
[4] El-Sayed AF. Aircraft propulsion and gas turbine engines. CRC Press; 2008.
[5] Haglind F. A review on the use of gas and steam turbine combined cycles as prime movers for large ships. Part I: background and design. Energ Conver Manage 2008;49:3458–67.
[6] Haglind F. A review on the use of gas and steam turbine combined cycles as prime movers for large ships. Part II: previous work and implications. Energ Conver Manage 2008;49:3468–75.
[7] Haglind F. A review on the use of gas and steam turbine combined cycles as prime movers for large ships. Part III: fuels and emissions. Energ Conver Manage 2008;49:3476–82.
[8] Lee S, Lee S, Kim K-H, Lee D-H, Kang Y-S, Rhee D-H. Optimization framework using surrogate model for aerodynamically improved 3D turbine blade design. In: Turbo Expo: Power for Land, Sea, and Air. American Society of Mechanical Engineers; 2014. p. V02BT45A019.
[9] Lee S-L, Shin S. Wind turbine blade optimal design considering multi-parameters and response surface method. Energies 2020;13:1639.
[10] Tang D, Tang B, Shen W, Zhu K, Quan Q, Deng Z. On genetic algorithm and artificial neural network combined optimization for a Mars rotorcraft blade. Acta Astronaut 2023;203:78–87.
[11] Wang Y, Liu T, Zhang D, Xie Y. Dual-convolutional neural network based aerodynamic prediction and multi-objective optimization of a compact turbine rotor. Aerosp Sci Technol 2021;116:106869.
[12] Li H, Song L, Li Y, Feng Z. 2D viscous aerodynamic shape design optimization for turbine blades based on adjoint method. J Turbomach 2010;133.
[13] Luo J, Xiong J, Liu F, McBean I. Three-dimensional aerodynamic design optimization of a turbine blade by using an adjoint method. J Turbomach 2010;133.
[14] Chan CM, Bai H, He D. Blade shape optimization of the Savonius wind turbine using a genetic algorithm. Appl Energy 2018;213:148–57.
[15] Wang L, Wang T-G, Luo Y. Improved non-dominated sorting genetic algorithm (NSGA)-II in multi-objective optimization studies of wind turbine blades. Appl Math Mech 2011;32:739–48.
[16] Yeo EJ, Kennedy DM, O'Rourke F. Tidal current turbine blade optimisation with improved blade element momentum theory and a non-dominated sorting genetic algorithm. Energy 2022;250:123720.
[17] Lee S-L, Shin SJ. Structural design optimization of a wind turbine blade using the genetic algorithm. Eng Optim 2022;54:2053–70.
[18] Oksüz OZ, Akmandor IBS. Axial turbine blade aerodynamic optimization using a novel multi-level genetic algorithm. In: Turbo expo: power for land sea, and air; 2008. p. 2361–74.
[19] Liao C, Zhao X, Xu J. Blade layers optimization of wind turbines using FAST and improved PSO algorithm. Renew Energy 2012;42:227–33.
[20] Li Y, Wei K, Yang W, Wang Q. Improving wind turbine blade based on multi-objective particle swarm optimization. Renew Energy 2020;161:525–42.
[21] Özkan R, Genç MS. Aerodynamic design and optimization of a small-scale wind turbine blade using a novel artificial bee colony algorithm based on blade element momentum (ABC-BEM) theory. Energ Conver Manage 2023;283:116937.
[22] Tahani M, Maeda T, Babayan N, Mehrnia S, Shadmehri M, Li Q, et al. Investigating the effect of geometrical parameters of an optimized wind turbine blade in turbulent flow. Energ Conver Manage 2017;153:71–82.
[23] Mokhtari Y, Rekioua D. High performance of maximum power point tracking using ant colony algorithm in wind turbine. Renew Energy 2018;126:1055–63.
[24] Uysal SC, Weiland N. Turbomachinery design of an axial turbine for a direct fired sCO2 cycle. Energ Conver Manage 2022;267:115913.
[25] Maleki A, Pourfayaz F, Rosen MA. A novel framework for optimal design of hybrid renewable energy-based autonomous energy systems: a case study for Namin, Iran. Energy 2016;98:168–80.
[26] Gunantara N. A review of multi-objective optimization: methods and its applications. Cogent Eng 2018;5:1502242.
[27] Jia L, Hao J, Hall J, Nejadkhaki HK, Wang G, Yan Y, et al. A reinforcement learning based blade twist angle distribution searching method for optimizing wind turbine energy power. Energy 2021;215:119148.
[28] Wang Z, Zeng T, Chu X, Xue D. Multi-objective deep reinforcement learning for optimal design of wind turbine blade. Renew Energy 2023.
[29] Fang J, Hu W, Liu Z, Chen W, Tan J, Jiang Z, et al. Wind turbine rotor speed design optimization considering rain erosion based on deep reinforcement learning. Renew Sustain Energy Rev 2022;168:112788.
[30] Deng Z, Xu C, Han X, Cheng Z, Xue F. Decentralized yaw optimization for maximizing wind farm production based on deep reinforcement learning. Energ Conver Manage 2023;286:117031.
[31] Zhang G, Hu W, Cao D, Zhang Z, Huang Q, Chen Z, et al. A multi-agent deep reinforcement learning approach enabled distributed energy management schedule for the coordinate control of multi-energy hub with gas, electricity, and freshwater. Energ Conver Manage 2022;255:115340.
[32] Zhang B, Hu W, Li J, Cao D, Huang R, Huang Q, et al. Dynamic energy conversion and management strategy for an integrated electricity and natural gas system with renewable energy: deep reinforcement learning approach. Energ Conver Manage 2020;220:113063.
[33] Zhang G, Hu W, Cao D, Liu W, Huang R, Huang Q, et al. Data-driven optimal energy management for a wind-solar-diesel-battery-reverse osmosis hybrid energy system using a deep reinforcement learning approach. Energ Conver Manage 2021;227:113608.
[34] Zhang B, Hu W, Cao D, Li T, Zhang Z, Chen Z, et al. Soft actor-critic–based multi-objective optimized energy conversion and management strategy for integrated energy systems with renewable energy. Energ Conver Manage 2021;243:114381.
[35] Neto G. From single-agent to multi-agent reinforcement learning: foundational concepts and methods. Learn Theory Course 2005;2.

[36] Zhang K, Yang Z, Başar T. Multi-agent reinforcement learning: a selective overview of theories and algorithms. In: Handbook of reinforcement learning and control; 2021. p. 321–84.
[37] Sutton RS, McAllester D, Singh S, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. Adv Neural Inf Proces Syst 1999;12.
[38] Konda V, Tsitsiklis J. Actor-critic algorithms. Adv Neural Inf Proces Syst 1999;12.
[39] J. Schulman, Levine S, Abbeel P, Jordan M, Moritz P. Trust region policy optimization. International conference on machine learning. PMLR; 2015. p. 1889–97.
[40] Watkins CJ, Dayan P. Q-learning. Mach Learn 1992;8:279–92.
[41] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature 2015;518:529–33.
[42] Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N. Dueling network architectures for deep reinforcement learning. In: International conference on machine learning. PMLR; 2016. p. 1995–2003.
[43] Azzouz R, Bechikh S, Ben Said L. Dynamic multi-objective optimization using evolutionary algorithms: a survey. In: Recent advances in evolutionary multi-objective optimization; 2017. p. 31–70.
[44] Liu R, Li J, Mu C, Jiao L. A coevolutionary technique based on multi-swarm particle swarm optimization for dynamic multi-objective optimization. Eur J Oper Res 2017;261:1028–51.
[45] Azzouz R, Bechikh S, Said LB. A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population management strategy. Soft Comput 2017;21:885–906.
[46] Deb K, Rao UB, Karthik S. Dynamic multi-objective optimization and decision-making using modified NSGA-II: a case study on hydro-thermal power scheduling. In: Evolutionary multi-criterion optimization: 4th international conference, EMO 2007, Matsushima, Japan, March 5–8, 2007 Proceedings 4. Springer; 2007. p. 803–17.
[47] Azzouz R, Bechikh S, Ben Said L. Multi-objective optimization with dynamic constraints and objectives: new challenges for evolutionary algorithms. In: Proceedings of the 2015 annual conference on genetic and evolutionary computation; 2015. p. 615–22.
[48] H. Chen, M. Li, X. Chen. Using diversity as an additional-objective in dynamic multi-objective optimization algorithms. In: 2009 Second international symposium on electronic commerce and security. IEEE; 2009. p. 484–7.
[49] Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, Part III 18. Springer; 2015. p. 234–41.
[50] Oktay O, Schlemper J, Folgoc LL, Lee, Heinrich M, Misawa K, et al. Attention u-net: Learning where to look for the pancreas. arXiv preprint arXiv:180403999; 2018.
[51] Çiçek Ö, Abdulkadir A, Lienkamp SS, Brox T, Ronneberger O. 3D U-Net: learning dense volumetric segmentation from sparse annotation. In: Medical image computing and computer-assisted intervention–MICCAI 2016: 19th international conference, Athens, Greece, October 17–21, 2016, Proceedings, Part II 19. Springer; 2016. p. 424–32.
[52] Jin X, Xi X, Zhou D, Ren X, Yang J, Jiang Q. An unsupervised multi-focus image fusion method based on Transformer and U-Net. IET Image Proc 2023;17:733–46.
[53] Zou Z, Wang S, Liu H, Yang C, Zhang W. Turbine aerodynamics for aero-engine: flow analysis and aerodynamics design. Shanghai, China: Shanghai Jiao Tong University Press; 2014.
[54] Shalev-Shwartz S, Shammah S, Shashua A. Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:161003295; 2016.
[55] Zhou M, Luo J, Villella J, Yang Y, Rusu D, Miao J, et al. Smarts: scalable multi-agent reinforcement learning training school for autonomous driving. arXiv preprint arXiv:201009776; 2020.
[56] Charbonnier F, Morstyn T, McCulloch MD. Scalable multi-agent reinforcement learning for distributed control of residential energy flexibility. Appl Energy 2022; 314:118825.
[57] Zhang B, Hu W, Ghias AM, Xu X, Chen Z. Multi-agent deep reinforcement learning based distributed control architecture for interconnected multi-energy microgrid energy management and optimization. Energ Conver Manage 2023;277:116647.
[58] Trigg MA, Tubby GR, Sheard AG. Automatic genetic optimization approach to two-dimensional blade profile design for steam turbines. J Turbomach 1999;121:11–7.
[59] Alexeev R, Tishchenko V, Gribin V, Gavrilov IY. Turbine blade profile design method based on Bezier curves. J Phys: Conf Ser 2017:012254.
[60] Hamakhan I, Korakianitis T. Aerodynamic performance effects of leading-edge geometry in gas-turbine blades. Appl Energy 2010;87:1591–601.
[61] Chibli HA, Abdelfattah SA, Schobeiri MT, Kang C. An experimental and numerical study of the effects of flow incidence angles on the performance of a stator blade cascade of a high pressure steam turbine. In: ASME turbo expo 2009: power for land, sea, and air; 2009. p. 821–30.
[62] Kiran K, Anish S. An investigation on the effect of pitchwise endwall design in a turbine cascade at different incidence angles. Aerosp Sci Technol 2017;71:382–91.
[63] Chen Y, Cai L, Jiang D, Li Y, Du Z, Wang S. Experimental and numerical investigations for turbine aerodynamic performance with different pressure side squealers and incidence angles. Aerosp Sci Technol 2023;136:108234.
[64] Popovic I, Zhu J, Dai W, Sjolander S, Praisner T, Grover E. Aerodynamics of a family of three highly loaded low-pressure turbine airfoils: measured effects of Reynolds number and turbulence intensity in steady flow. In: Turbo expo: power for land, sea, and air; 2006. p. 961–9.
[65] McKay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 2000;42:55–61.
[66] Stein M. Large sample properties of simulations using Latin hypercube sampling. Technometrics 1987;29:143–51.
[67] Iooss B, Lemaître P. A review on global sensitivity analysis methods. uncertainty management in simulation-optimization of complex systems: algorithms and applications; 2015. p. 101–22.
[68] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Proces Syst 2019;32.
[69] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:170706347; 2017.
[70] Guan Y, Ren Y, Li SE, Sun Q, Luo L, Li K. Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization. IEEE Trans Veh Technol 2020;69:12597–608.
[71] Bøhn E, Coates EM, Moe S, Johansen TA. Deep reinforcement learning attitude control of fixed-wing UAVS using proximal policy optimization. In: 2019 International conference on unmanned aircraft systems (ICUAS). IEEE; 2019. p. 523–33.
[72] Xue D, Wu D, Yamashita AS, Li Z. Proximal policy optimization with reciprocal velocity obstacle based collision avoidance path planning for multi-unmanned surface vehicles. Ocean Eng 2023;273:114005.
[73] Sutton RS. Learning to predict by the methods of temporal differences. Mach Learn 1988;3:9–44.
[74] Yu C, Velu A, Vinitsky E, Gao J, Wang Y, Bayen A, et al. The surprising effectiveness of PPO in cooperative multi-agent games. Adv Neural Inf Proces Syst 2022;35:24611–24.
[75] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. Openai gym. arXiv preprint arXiv:160601540; 2016.
[76] Fonseca CM, Paquete L, López-Ibánez M. An improved dimension-sweep algorithm for the hypervolume indicator. In: 2006 IEEE international conference on evolutionary computation. IEEE; 2006. p. 1157–63.
[77] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6:182–97.
[78] Verma S, Pant M, Snasel V. A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems. IEEE Access 2021;9:57757–91.