Reinforcement Learning-Based Wind Farm Control: Towards Large Farm Applications via Automatic Grouping and Transfer Learning

Hongyang Dong and Xiaowei Zhao

***Abstract*—The high system complexity and strong wake effects bring significant challenges to wind farm operations. Conventional wind farm control methods may lead to degraded power generation efficiency. A reinforcement learning (RL)-based approach is proposed in this paper to handle these issues, which can increase the long-term farm-level power generation subject to strong wake effects while without requiring analytical wind farm models. The proposed method is significantly distinct from existing RL-based wind farm control approaches, whose computational complexities usually increase heavily with the increase of total turbine numbers. In contrast, our method can greatly reduce training loads and enhance learning efficiency via two novel designs: (1) automatic grouping and (2) multi-agent-based transfer learning (MATL). Automatic Grouping can divide a large wind farm into small turbine groups by analyzing the aerodynamic interactions between turbines and utilizing some key principles from the graph theory. It enables the separated conduction of RL algorithms on small turbine groups, avoiding the complex training process and high computational costs of applying RL on the entire farm. Based on Automatic Grouping, MATL can further reduce the computational complexity by allowing agents (i.e. wind turbines) to inherit control policies under potential group changes. Case studies with a dynamical simulator show that the proposed method achieves clear power generation increases than the benchmark. It also dramatically reduces computational costs compared with typical RL-based wind farm control methods, paving the way for the application of RL in general wind farms.**

***Index Terms*—Reinforcement learning, wind farm control, machine learning, wind energy, intelligent control.**

I. Introduction

Wind energy is essential for the global effect in achieving a resilient green transition, and it is now helping the world avoid over 1.2 billion tonnes of CO2 annually [1]. It is the primary and dominant renewable energy – the power generated by wind is almost as much as the combination of all the other renewable sources. The wind industry has been growing dramatically in recent years, and many new wind farms have been planned globally to meet the needs of net zero. Among all the sub-systems, the wind farm control/operation system is essential and directly decides wind farms' power generation process and economic profitability [2]. With the continuous increase of global wind energy capacity, operating wind farms more efficiently and economically has become a key need in the wind industry, especially for large offshore wind farms. However, the high system complexity and strong aerodynamic interactions between turbines render the design of wind farm control systems challenging. Particularly, it is well-understood and widely reported [2], [3], [4], [5] that the power generation efficiency of downstream turbines in a wind farm can be severely affected by the wakes (with reduced wind speeds and increased turbulence levels compared to the free stream) induced by upstream turbines. Currently, most wind farms ignore such wake effects in practical operation. They usually set all turbines to work under the local greedy control strategy – all turbines focus on maximizing their own power generation while omitting any potential influences on other turbines. Such a control strategy is easy to implement but can be far away from globally optimal, as demonstrated in a lot of works, e.g., Refs. [6], [7], [8]. Many recent studies have shown that, by controlling all turbines in the farm via cooperative patterns, wake effects can be potentially alleviated and the whole farm's total power generation can be increased compared with the cases under the greedy strategy. Site test results in Refs. [9], [10] proved this fact, showing that advanced wind farm control methods are needed to optimize farm-level power generation.

A challenge in designing cooperative wind farm controllers is modelling the interactions of wind farms with their flow fields, which usually requires solving complex PDE equations, e.g., the Navier-Stokes equations. The stochastic natures of flow fields and wakes further increase the modelling tasks' difficulty. An intuitive way to handle these issues is building or employing simplified models and carrying out optimization/control based on them. Some notable studies in this direction are given in Refs. [6], [11], [12], [13], [14], and typical methods include game-theoretic algorithm [6], [13], Bayesian-based algorithm [12], particle swarm optimization [11], random search [14], etc. However, most of these elegant results rely on steady-state models or data collected/predicted by steady-state models. The limited fidelity of steady-state models may degrade these methods' performance in practical applications. In addition, achieving closed-loop adjustments under time-varying environments is another challenge that potentially decreases the applicability of these results. Model predictive control (MPC) methods are proposed in [7], [15], [16] to provide potential solutions for these issues. They are able to carry out closed-loop wind farm control under dynamic wind farm models/data and achieve better performance compared to steady-state optimization. Nevertheless, these elegant

This work was funded by the UK Engineering and Physical Sciences Research Council under Grant EP/S000747/1.

H. Dong and X. Zhao (corresponding author) are with the Intelligent Control and Smart Energy (ICSE) Research Group, School of Engineering, University of Warwick, Coventry CV4 7AL, UK. Emails: hongyang.dong@warwick.ac.uk, xiaowei.zhao@warwick.ac.uk.

results still highly rely on the accuracy of underlying wind farm models (including wind flow and wake models), and their performance can be influenced in the presence of inevitable modelling errors and unmodelled dynamics.

The wind industry requires innovations to deal with these limitations in wind farm control, for which reinforcement learning (RL) has been proven to be a promising and effective candidate. RL is a booming interdisciplinary technology that has been developed dramatically in recent years [17], [18], [19]. It can handle many complex tasks that are almost impossible to be addressed by conventional control methods, such as beating professional human players in the board game GO [18]. RL methods have been applied to many areas, such as robotics [20], [21], aerospace engineering [22], and so on. One of the key ideas of RL is capturing the essential system information and improving control policies via interacting with the environment, allowing RL algorithms to handle optimal control problems of black-box systems (i.e. systems with unknown models). Applying RL in wind farm operations has become a cutting-edge research area, and its feasibility has been proved in recent studies [23], [24], [25], [26], [27], [28], [29], [30], [31]. For example, a model-free approach for wind farm power optimization was introduced in [23] via the deep Q-network algorithm [32]. Ref. [24] employed physical models to guide RL training and achieved wind farm power maximization. Ref. [28] designed a robust RL method for farm-level power tracking tasks. These results demonstrate that RL algorithms can achieve complex wind farm control tasks subject to uncertain/unknown system models, showing strong robustness and adaptability compared with conventional wind farm control methods.

However, several issues can hinder the application of RL-based wind farm control methods. Specifically, the computational complexity of these algorithms increases exponentially with the increase of action & state dimensions. Though distributed structures are employed in some studies, they still need to evaluate the long-term reward of the whole farm (such as the long-term farm-level power generation), which still requires the measurements and/or states of all turbines. The training/learning stability, on the other hand, deteriorates under large action & state spaces. Therefore, the computational loads of existing RL-based wind farm control methods can grow significantly with the increase of turbine numbers, while the efficiency & stability of the training process are degraded.

All these facts motivate us to design new wind farm control methods to overcome the limitations of current studies. This paper develops an application-oriented RL method for wind farm power maximization tasks subject to time-varying environmental conditions and strong aerodynamic interactions between turbines. We build upon a multi-agent actor-critic structure. Specifically, each agent, i.e. each turbine in the farm, employs an actor network to learn its own control policy, while critic networks are designed to evaluate the long-term reward functionals of special turbine groups. Deep neural networks (DNNs) are employed in our design as information processors, and the actor & critic DNNs are trained via the proximal policy optimization (PPO) algorithm [33], [34] with only measurable data at turbine rotors (instead of full states of the whole flow field). In addition, two novel designs in our RL-based wind farm control method are described in detail as follows.

(I) Automatic Grouping (AG). AG is designed to significantly reduce the proposed method's overall computational load, especially for large wind farm applications. It can divide the whole wind farm into small turbine groups by analyzing the aerodynamic interactions between turbines and employing some fundamental ideas from the graph theory [35].

(II) Multi-Agent-based Transfer Learning (MATL). Based on AG and some key properties of wind farm control problems, MATL allows agents (i.e. turbines) to inherit the training results of their actor DNNs as the initial settings under potential grouping changes. Moreover, MATL also carries out supervised-learning-style pre-training for critic DNNs of newly formed groups.

In addition to these designs, some other key contributions of the proposed method are summarised as follows.

(1) This paper develops a new reinforcement learning-based wind farm control method that can overcome the limitations of many existing methods, including high reliance on system models, lack of robustness to stochastic environmental conditions, and need for immeasurable data. The proposed method is capable of achieving data-driven closed-loop control for farm-level power maximization under time-vary wind speeds & directions with only practically available measurements.

(2) The proposed method can significantly mitigate the high computational loads & training costs associated with almost all deep RL-based wind farm control approaches. Employing AG and MATL allows users to execute RL on small turbine groups, avoiding carrying out the complex training with the whole farm. Moreover, AG also reduces the sizes of action & state spaces, potentially improving training efficiency and stability. All these enhance the applicability and scalability of the proposed method to general and large wind farms.

(3) The proposed method has a general framework. AG and MATL are designed to be "plug-and-play" modules to the main RL structure. They can be applied to not only our RL algorithm but also other RL-based wind farm control methods. Moreover, given the data-driven feature, the proposed method does not restrict the types of turbines and induction control inputs - it allows different induction states, e.g., induction factors and/or thrust coefficients, to be employed in different scenarios.

We employ a dynamic wind farm simulator introduced in [36] in case studies and consider yaw angles & thrust coefficients as control inputs to test the performance and advantages of the proposed method. Nevertheless, the proposed method can be applied to other simulators or real wind farms and employ other relevant states as control inputs.

It is noteworthy that this paper is partially built upon the conference version in [30], while significant new contributions have been made from the following several aspects: (1) Based on the adjacency matrix and a restricted depth-first search approach, a solid method (Algorithms 1 & 2) is proposed in this paper to generate turbine groups automatically - to our best knowledge, this is for the first time an automatic grouping strategy is proposed for wind farm operational tasks. In contrast, Ref. [30] requires manual grouping. (2) Significantly different from [30], a new RL-based wind farm control method is developed in this paper via multi-agent proximal policy optimization. Compared with [30], this new method is more suitable for handling wind farm control problems, especially in cooperation with the automatic grouping algorithm. (3) As a new design, the transfer learning strategy is employed in this paper. It allows the proposed method to handle potential grouping changes due to wind direction variations and reduce training loads further.

In the remainder of this paper, the wind farm control problem is introduced in Sec. II. Our RL-based control method's design details are given in Sec. III. Case studies are demonstrated in Sec. IV to analyze the performance and advantages of the proposed method. Finally, this paper ends with some conclusive remarks in Sec. V.

II. Problem Formulation

We consider a wind farm with n turbines to formulize the control problem considered in this paper, and we use $\mathcal{WT}_1$, $\mathcal{WT}_2$, ..., $\mathcal{WT}_n$ to denote all the turbines. Without loss of generality, for a turbine i (denoted as $\mathcal{WT}_i$) in the farm, its power generation P_i can be described by the following equation.

$$P_i = F_i(U_i, \alpha_i, \gamma_i) \tag{1}$$

where U_i denotes the wind speed at the rotor of $\mathcal{WT}_i$, α_i denotes the induction state (such as the induction factor or other relevant states, e.g. the thrust coefficient), and γ_i is the yaw angle. F_i expresses the relationship of U_i, α_i, γ_i with P_i. The RL method that will be developed in the following section is data-driven and does not require the specific expression of F_i. In other words, F_i can be unknown for the proposed method in this paper.

The primary goal of wind farm control is to maximize farm-level power generation from a long-term point of view. Other requirements should also be considered, such as avoiding large structural loads and limiting control actions. Based on that and following relevant studies [37], [24], [26], we consider a reward/performance metric that can balance the power generation and structural loads:

$$J = \sum_{t=0}^{\infty}\sum_{i=1}^{n} \xi^t C_i(t) \tag{2}$$

with

$$C_i(t) = w_1 P_i(t) - w_2 L_{\alpha_i}(t) - w_3 L_{\gamma_i}(t) \tag{3}$$

Here t denotes the time step, $0 < \xi \leq 1$ is a discount factor, and w_1, w_2 and w_3 are user-defined constants for weighting purposes. The terms in C_i are explained as follows.

(1) The first term $w_1 P_i(t)$ in Eq. (3) is for farm-level power generation maximization - the larger it is, the higher the whole farm's power generation.

(2) The term $w_2 L_{\alpha_i}(t)$ in Eq. (3) is to consider loads caused by wind farm control. We set

$$L_{\alpha_i}(t) = |f_i(t) - f_i(t-1)| \tag{4}$$

with $L_{\alpha_i}(0) = 0$. Here f_i is the axial force that the inflow wind exerts on Turbine i, described by [36], [37]

$$f_i = \frac{1}{2}\rho A_i U_i^2 \cos^2(\gamma_i) c'_{T_i} \tag{5}$$

where ρ is the air density, A_i is the rotor plane's swept area, and c'_{T_i} is called the modified thrust coefficient [36], [37]. Based on Ref. [37], L_{α_i} is related to dynamical turbine loading. Therefore, introducing L_{α_i} into the performance metric allows us to balance dynamical loads and power generation.

(3) The third term $L_{\gamma_i}(t)$ is to avoid large yaw offsets and yaw-induced structural loads, and we set $L_{\gamma_i}(t) = |\gamma_i(t)|$.

Based on Eqs. (2) and (3), our objective is controlling α_i and γ_i to maximize J. This optimal control problem is challenging due to the high system complexity and the wake effect (as qualitatively illustrated in Fig. 1.(a)) associated with wind farm control tasks. To be specific, many studies [6], [38], [28] and real-world applications have shown that the wake induced by an upstream wind turbine (with decreased wind speed and increased turbulence level) can severely influence the power generation of its downstream turbines. The traditional greedy strategy (in which each turbine in a wind farm only focuses on maximizing its own power generation while ignoring its influence on other turbines due to wake effects) can be far from optimal. Therefore, farm-level control strategies should be considered to operate all turbines cooperatively in order to mitigate wake effects and increase the whole farm's power generation. Moreover, wake effects commonly have time-delayed features - it usually takes tens or hundreds of seconds for a wake to propagate from an upstream turbine to a downstream turbine. This means the wake generated by an upstream turbine can influence the future power generation of its downstream turbines. Thus wind farm control strategies should not be short-sighted. It should be able to optimize the long-term farm-level reward as given in (2). This paper employs a novel RL strategy to achieve this goal.

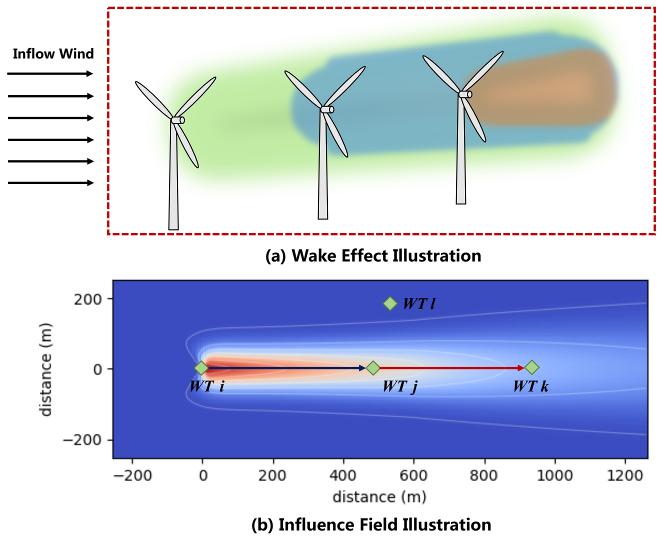

Fig. 1: Illustrations of wake effect and influence field.

III. Reinforcement Learning-Based Wind Farm Control

This section proposes an RL-based wind farm method to maximize the long-term reward. As introduced in Sec. I, this new method aims to overcome the limitations of conventional methods and also current RL-based approaches. It contains three key modules: (1) An automatic grouping approach; (2) An RL structure via multi-agent proximal policy optimization; (3) A multi-agent-based transfer learning strategy. Their specific designs are given in the following subsections.

A. Automatic Grouping

Automatic Grouping (AG) aims to reduce the overall computational loads of our RL-based wind farm control method by separating the wind farm into turbine groups. There are several important observations & principles that guide the design of AG:

(a) The influence of wake decays as it propagates. Therefore, it is important to quantitatively analyze aerodynamic interactions based on relative turbine positions.

(b) If the AG algorithm is too conservative and takes into account all potential interactions between turbines, then some groups can be large and contain many turbines, especially considering the cascading effects. To mitigate this issue, AG should justify the level of aerodynamic interactions and potentially restrain the number of turbines in each group.

(c) It is noteworthy that for large wind farms, the most downstream turbines in one group can be set as the most upstream turbines in the following group along the inflow wind direction. Such a design can help the AG algorithm take into account the cascading effects and therefore fully reflect the aerodynamic interactions between different groups.

Based on these observations and principles, the essential steps in the AG strategy are introduced as follows.

Step 1: Calculating the Influence Factor. Here we define a concept called the influence factor. It is a quantitative way to show the influence of a single turbine's power generating process on the flow field around it. For any spatial position H in the farm, we use $\mathcal{IF}_{WT_i \to H}$ to denote the influence factor of turbine $\mathcal{WT}_i$ with respect to the position H. Following [30], the specific definition of $\mathcal{IF}_{WT_i \to H}$ is

$$\mathcal{IF}_{WT_i \to H} = \frac{1}{\gamma_{\max} - \gamma_{\min}} \int_{\gamma_{\min}}^{\gamma_{\max}} w_\gamma(\gamma_i) \cdot \delta U_H(\gamma_i) \mathrm{d}\gamma_i \quad (6)$$

In Eq. (6), $\gamma_{\min}$ and $\gamma_{\max}$ are the maximum and minimum allowable yaw offsets of $\mathcal{WT}_i$ with respect to the wind direction, $w_\gamma(\gamma_i)$ is a non-negative function of the yaw angle γ_i, providing the freedom for weighting aims. Moreover, $\delta U_{H_j}(\gamma_i)$ is the wind speed deficit rate when α_i is under the greedy strategy and the yaw offset w.r.t the wind direction is set to be γ_i. Therefore, $\mathcal{IF}$ is essentially a weighted integral of the deficit rates under different yaw offsets. It is noteworthy that, compared with induction states, yaw angles have critical influences on the direction of wakes and play a key role in wake steering. These facts motivate us to employ the yaw offset in the definition of $\mathcal{IF}$. It should also be emphasized that only essential wake features are needed to carry out AG. Therefore, the calculation of $\delta U_H(\gamma_i)$ does not require high-fidelity flow field models, and analytical steady-state models already can satisfy the needs. As an example, Fig. 1.(b) illustrates the distribution of influence factor of a turbine $\mathcal{WF}_i$, in which the values of $\mathcal{IF}_{WT_i \to H}$ (H indicates any position) are calculated by the FLORIS model proposed in [6] under the conditions $\gamma_{\min} = -30°$, $\gamma_{\max} = 30°$ and $w_\gamma(\gamma_i) \equiv 0$. In this figure, the darker the color, the higher the value of $\mathcal{IF}_{WT_i \to H}$. One can see that the wake induced by WT_i has severe potential influences on WT_j, while its influences on WT_k and WT_l are at a much lower level.

Step 2: Getting the Influence Field by $\mathcal{IF}$. The $\mathcal{IF}$ value defined in (6) can quantitatively show a turbine's influence on the flow field. By setting a threshold (denoted as b_{IF}) to $\mathcal{IF}$, one can further get an influence field of any turbine in the farm, as shown by the envelopes (which indicate the edges of influence fields under different thresholds) in Fig. 1.(b). To bring flexibility for AG, b_{IF} is set to be a user-defined parameter. After setting b_{IF}, for a turbine $\mathcal{WT}_i$, any other turbines that are within its influence field are said to have significant aerodynamic interactions with $\mathcal{WT}_i$.

Step 3: Building a Graph Based on Turbines' Interactions. We employ some fundamental ideas from the graph theory [35] in the design of AG. Particularly, we treat each turbine in the farm as a vertex in a graph. Suppose a downstream turbine is within the influence field of an upstream turbine. In that case, a directed edge from the upstream turbine to the downstream turbine should be added to the graph, indicating that the corresponding turbines have significant aerodynamic interactions. Following that, all turbines' interactions can be described by a directed graph.

Step 4: Restricting the Depth of the Graph to Get Grouping Results. As we discussed before, turbines may have cascading interactions. We take Fig. 1.(b) as an example. If b_{IF} is set to be 0.3, WT_k is outside the influence field of WT_i. But, WT_k is within the influence filed of WT_j, so WT_k and WT_i are connected via WT_j in the graph, rendering these three turbines to be in the same group if no restrictions are introduced. For large wind farms with long turbine rows (along the wind direction), such cascading aerodynamic interactions may lead to large turbine groups, which can be undesirable. In our AG algorithm, the depth of the graph can be restricted to break down large turbine groups into small sub-groups. Moreover, the most downstream turbines in one group can be set as the most upstream turbines in the following group. This allows the algorithm to still consider the cascading effects in the grouping. The final grouping results can be obtained by carrying out a restricted depth-first search (RDFS) (modified from DFS [35]) to get connected components.

Based on these designs and analyses, the automatic grouping (AG) algorithm is summarized in Algorithm 1, and a restricted depth-first search (RDFS) algorithm (which is employed in Algorithm 1) is given in Algorithm 2. Please note "#" indicates comments in Algorithms 1 and 2. Please note a recursive function is employed in RDFS, as shown in Algorithm 2. Particularly, the IF loop (Lines 6-8) in Part II of Algorithm 2 is key to the stop logic of the recursive function. Once the corresponding vertex has been visited or its depth is beyond the user-defined limit, the IF condition cannot be satisfied and no further recursive function for that vertex is carried out. This, together with the visit-labelling law in Line 2 and the depth-updating law in Line 5, forms the stop criteria of our recursive function.

Algorithm 1 Automatic Grouping (AG) Algorithm.

Input: turbine labels and positions.

Output: lists of turbine labels - each list indicates a group of turbines.

```
1:  Set the threshold for $\mathcal{IF}$ (denoted by $b_{IF}$) and the depth limit for grouping (denoted by $b_d$).
2:  Define a graph for the wind farm by an adjacency matrix $G[n][n]$, with $n$ being the total turbine number. Each turbine $\mathcal{WT}_i$ is a vertex of the graph, denoted by $v_i$.
3:  All entries in $G[n][n]$ are initialized to be zero.
4:  Mark the most upstream turbines as root vertices.
    # Initialization ends
5:  **for** $i = 0$ to $n$ **do**
6:      **for** $j \neq i$ **do**
7:          Calculate the influence factor $\mathcal{IF}_{WT_i \to WT_j}$ via (6).
8:          **if** $\mathcal{IF}_{WT_i \to WT_j} \geq b_{IF}$ **then**
9:              Add a directed edge to the graph, i.e. set $G[v_i][v_j] = 1$.
10:         **end if**
11:     **end for**
12: **end for**
13: **while** the adjacency matrix $G$ is not empty **do**
14:     Based on the adjacency matrix $G$, carry out the restricted depth-first search (RDFS) from root vertices with the maximum depth being $b_d$. See Algorithm 2 for more details.
15:     Based on the result of RDFS, collect the vertices (i.e. the turbine labels) of every unique connected component and store it in a list as a turbine group.
16:     Mark the vertices with depths being $b_d$ as root vertices.
17:     Remove the vertices that have been searched with depths less than $b_d$ from $G$.
18: **end while**
19: **return** lists of turbine labels.
```

Remark 1 (selection of b_{IF} and b_d): The parameters b_{IF} and b_d are important in the AG algorithm. Their values are set to be user-defined to meet different needs since they can vary the grouping results. For example, if one sets $b_{IF} = 0.3$ for the case given in Fig. 1.(b), then the influence field is defined by $\mathcal{IF}_{WT_i \to H} \geq 0.3$, and WT_j is within this influence field while WT_k & WT_l are outside. One can see b_{IF} can decide the size of the influence field (as shown by the envelopes in Fig. 1.(b), which are the edges of different influence fields under different values of b_{IF}) - the smaller the b_{IF}, the larger the influence area, and vice versa. AG allows users to decide b_{IF} based on their own needs, therefore adapting to different conditions and scenarios. As a quick guidance, [0.15, 0.4] is a proper interval for setting b_{IF} - this is based on the definition in (6) (which is essentially a weighted wind speed deficit rate) and the observations from extensive simulation results.

The parameter b_d decides the maximum depth of each individual connected component in the implementation of AG and therefore avoids large turbine groups due to the cascaded aerodynamic interactions between turbines. It must be a non-zero integer. In addition, setting $b_d = 0$ will lead to only single-turbine groups. Therefore, one should follow $b_d \in \mathbb{N}^+$ and decide the value of b_d based on their specific needs. As explained in Step 4 in AG design, our algorithm can consider cascading effects. Thus, a very large b_d is unnecessary and setting $b_d \in \{2, 3, 4, 5\}$ is reasonable and suggested.

Algorithm 2 Restricted Depth-First Search (RDFS).

I. Implementation

```
1:  **RDFS**$(G, RV\_list)${
2:  # $G$: adjacency matrix; $RV\_list$: list of root vertices.
3:  Initialize a list to store results: $Group\_list$ = [].
4:  **for** each vertex $v_i$ in $G$ **do**
5:      Initialize a data structure for $v_i$: $\mathcal{V}_i = \{$ $depth = 0$; $visited =$ false; $label = v_i\}$.
6:  **end for**
7:  **for** each root vertex in $RV\_list$ (denoted by $v_r$) **do**
8:      **RF**$(G, \mathcal{V}_r, b_d)$    # as defined in Part II below.
9:  **end for**
10: **for** each vertex $v_k$ in $Group\_list$ **do**
11:     **if** $\mathcal{V}_k.depth > b_d$ **then**
12:         $Group\_list.remove(\mathcal{V}_k.label)$
13:     **end if**
14: **end for**
15: **return** $Group\_list$}
```

II. Recursive Function

```
1:  **RF**$(G, \mathcal{V}_r, b_d)${
2:  $\mathcal{V}_r.visited$ = true    # indicate $\mathcal{V}_r$ has been visited.
3:  $Group\_list.append(\mathcal{V}_r.label)$
    # store the corresponding turbine label (i.e. the vertex label of $\mathcal{V}_r$) into the list.
4:  **for** each $\mathcal{V}_j$ such that $G[\mathcal{V}_r.label][\mathcal{V}_j.label] == 1$ **do**
5:      $\mathcal{V}_j.depth = \max\{\mathcal{V}_r.depth + 1, \mathcal{V}_j.depth\}$
        # update the depth of $\mathcal{V}_j$.
6:      **if** ($\mathcal{V}_j.depth \leq b_d$) and ($\mathcal{V}_j.visited ==$ false) **then**
7:          **RF**$(G, \mathcal{V}_j, b_d)$
8:      **end if**
9:  **end for**}
```

B. Multi-Agent Reinforcement Learning

This subsection develops the RL-based wind farm controller to be applied to each turbine group from the result of AG. We set the total number of turbine groups from AG to be L.

Without loss of generality, we consider a group (denoted by $\mathcal{TG}_l$) with a total of m turbines. Each turbine in the group is regarded as an agent. Therefore, the control problem of this turbine group can be transformed into a cooperative game with m agents, and the core objective is to design control policies for all the turbines in the group to maximize the following cumulative discounted reward from the time step t:

$$R_t = \sum_{\mathcal{WT}_i \in \mathcal{TG}_l} \sum_{h=0}^{\infty} \xi^{t+h} C_i(t+h) \tag{7}$$

where C_i follows the definition in (2).

We build our RL structure upon the multi-agent proximal policy optimization (PPO) algorithm [34]. Particularly, we employ a multi-agent actor-critic structure. Each agent employs an actor DNN (denoted by π_i, parameterized by θ_i) to learn its control policy, and there is a central critic DNN (denoted by V_ϕ, parameterized by ϕ) to learn the central state-value function (which is defined based on (7)):

$$V_\pi(s(t)) = \mathbb{E}[\sum_{\mathcal{WT}_i \in \mathcal{TG}_l} \sum_{h=0}^{\infty} \xi^{t+h} C_i(s(t+h), \pi_\theta)] \quad (8)$$

where $s = \{s_1, s_2, ..., s_m\}$ and $\pi_\theta = \{\pi_1, \pi_2, ..., \pi_m\}$ denote the aggregated state and control policy of all agents, respectively. Here s_i is the state of agent i, with $i = 1, 2..., m$, and $\theta = \{\theta_1, \theta_2, ..., \theta_m\}$. One can see $V_\pi(s(t))$ is actually the cumulative reward from time t, with the initial aggregated state to be $s(t)$ and the control policy to be π_θ at time t and thereafter. Based on (8), the optimal control strategy can be described by

$$\pi^* = \arg\max_\pi V_\pi(s(t)) \quad (9)$$

It is essential to decide the update laws for actor and critic DNNs to approximate $V_\pi(s)$ and π^*, and we employ the PPO method [33], [34] to achieve this aim. At each learning step, a set of trajectories (denoted by $\mathcal{D}$, with a size of N) is employed to carry out the update of θ_i and ϕ, and the corresponding losses, $L(\theta_i)$ and $L(\phi)$ are defined as follows.

$$L(\theta_i) = \frac{1}{NT} \sum_{\tau \in \mathcal{D}_\tau} \sum_{t=0}^{T} \min\{\frac{\pi_i(a_{i,t}|s_{i,t})}{\pi_{i,\text{old}}(a_{i,t}|s_{i,t})} A_{\pi_i,t}, g(\epsilon, A_{\pi_i,t})\} \quad (10)$$

$$L(\phi) = \frac{1}{NT} \sum_{\tau \in \mathcal{D}_\tau} \sum_{t=0}^{T} (V_\phi(s_t) - \hat{R}_t)^2 \quad (11)$$

where τ indicates the trajectories in the set D_τ, T is the length of each trajectory, $\pi_{i,old}$ is the old control policy of the agent i from the previous learning step, $a_{i,t}$ is the control action of the agent i at the time step t, $s_{i,t}$ is the state of the agent i at t, and $s_t = \{s_{1,t}, s_{2,t}, ..., s_{m,t}\}$. Moreover, $A_{\pi_i,t}$ is calculated by the general advantage estimation (GAE) [39], and the function $g(\epsilon, A_{\pi_i,t})$ is defined by

$$g(\epsilon, A_{\pi_i,t}) = \begin{cases} (1+\epsilon)A_{\pi_i,t} & \text{if } A_{\pi_i,t} \geq 0 \\ (1-\epsilon)A_{\pi_i,t} & \text{if } A_{\pi_i,t} < 0 \end{cases} \quad (12)$$

and here ϵ is a small user-defined constant which indicates how far away the new policy is allowed to go from the old [33]. In addition, $\hat{R}_t$ is the cumulative discounted reward from t to the end of the trajectory, serving as an estimate of R_t.

With $L(\theta_i)$ and $L(\phi)$, the actor and critic DNNs can be updated by maximizing $L(\theta_i)$ $(i = 1, 2, ..., m)$ and minimizing $L(\phi)$, respectively.

Then we are ready to mold the wind farm control problem into this multi-agent RL structure. For a turbine WT_i, its control signals (i.e. the outputs of π_i) are the changes of α_i and γ_i, denoted by $\delta\alpha_i$ and $\delta\gamma_i$, respectively. In the RL algorithm, the minimum and maximum values of α_i, γ_i, $\delta\alpha_i$, and $\delta\gamma_i$ are restricted based on turbine specifications. The function C_i that decides the reward is defined in (2), which is related to the turbine's power generation P_i, the yaw angle γ_i, and the change of the induction state $\delta\alpha_i$. The local observations (i.e. the inputs of π_i to decide control signals) include the power generation P_i, yaw angles γ_i, induction state α_i, wind speed at the rotor, wind direction, and turbine position.

Remark 2 (Addressing potential control policy conflicts): AG allows the proposed multi-agent RL method to be carried out in small turbine groups instead of the whole farm. As discussed before, that can significantly mitigate the high computation loads associated with deep RL and also enhance training efficiency and stability by reducing state & action spaces. To combine AG with multi-agent RL, we also need to address the potential control policy conflicts induced by the overlaps of different turbine groups (i.e. some turbines may be included in more than one group.) Based on the design of AG, only the most downstream turbines in one group have the possibility to be included in other groups. Given this important property, the RL method can directly set the most downstream turbines in a group to always follow the greedy strategy while learning the control policies for other turbines. This is also a common practice in real wind farms because the wakes induced by the most downstream turbines in a farm have ignorable influences on the farm-level power generation. But in our design, if these most downstream turbines are also contained in other groups as the most upstream turbines, their control policies should be decided by the learning results of those other groups. This logic helps the proposed wind farm control method address the potential control policy conflicts.

C. Multi-Agent Based Transfer Learning

The large change of free-flow wind directions (denoted by $\delta\Upsilon$) can vary the aerodynamic interactions between turbines and the result of AG. When applying the proposed wind farm control method, a threshold b_Υ can be set to refresh AG. If the new AG results do indicate the changes in some groups, updates need to be made to adapt to the new situations. Performing full re-training for the changed groups is feasible but clearly time-costly. Here we employ a multi-agent based transfer learning (MATL) strategy to smooth the re-training process of the proposed RL-based wind farm control method under AG changes.

The core idea of MATL is allowing agents (i.e. turbines) to inherit the learning results of their actor DNNs and use them as initial settings when their associated group is changed. In addition, with the inherited actor DNNs, MATL also carries out supervised-learning-style pre-training for the initilization of the central critic DNNs of the changed groups. Therefore, the full re-training is avoided, and only fine-tuning is needed for new groups. The key principles that build the feasibility for MATL are indicated in the proposed AG and multi-agent RL. In particular, the multi-agent RL in Sec. III.B has a centralized training decentralized execution (CTDE) structure. The actor DNN for each agent only needs local observations to get control signals, rendering it can be inherited after grouping changes.

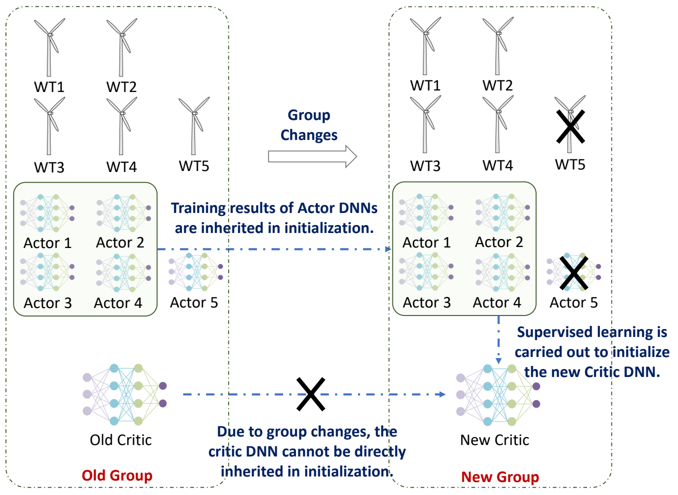
WT1
WT2
WT3
WT4
WT5
Group Changes
WT1
WT2
WT3
WT4
WT5
Actor 1
Actor 2
Actor 3
Actor 4
Actor 5
Training results of Actor DNNs are inherited in initialization.
Actor 1
Actor 2
Actor 3
Actor 4
Actor 5
Supervised learning is carried out to initialize the new Critic DNN.
Old Critic
Due to group changes, the critic DNN cannot be directly inherited in initialization.
New Critic
Old Group
New Group

Fig. 2: An example of MATL.

An example showing the implementation procedure of MATL is illustrated in Fig. 2. Assume there are five turbines in a group, then AG updates and $\mathcal{WT}_5$ is removed from this group. Following the MATL rule, all the new group's turbines inherit their actor DNN training results from the old group. Such an initialization process can help the RL's training and fine-tuning for the new group. However, we cannot do the same thing directly for the new central critic because the critic now aims to optimize the power generation of the new group of turbines. Instead, the initialization of the new critic DNN is carried out by supervised learning based on actor DNNs. It should be emphasized that MATL targets the initialization process of RL re-training after group changes. It allows the RL algorithm to make a good initial guess based on the results of trained DNNs (especially under mild main wind direction variations), potentially enhancing learning effectiveness and reducing training loads. But fine-tuning is still required after group changes, particularly for situations with significant wind direction changes.

Remark 3: The main structures of the proposed RL-based wind farm control method are illustrated in Fig. 3. We summarize connections between different parts and the main implementation steps (as indicated in Fig. 3) of our method in this remark. (1) Firstly, the AG algorithm (Algorithm 1) proposed in Sec. III.A can divide the whole wind farm into small turbine groups. (2) The multi-agent RL algorithm in Sec. III.B takes the result from AG. Then, based on the measurements and observations from the wind farm, our RL algorithm aims to optimize the total power generation of each turbine group. It employs DNNs to form a multi-actor central-critic structure, and the corresponding update laws are provided in Sec. III.B in detail. (3) Each turbine in the farm receives control commands from its associated actor DNN and provides measurements to the RL algorithm. (4) As discussed in Sec. III.C, the AG result will be refreshed if the variation of main wind direction exceeds the user-defined threshold b_{Υ}. The multi-agent based transfer learning (MATL) strategy is triggered only when the AG result changes. As designed in Sec. III.C, MATL allows the inheritance of DNN training results in the initialization of re-training/fine-tuning.

IV. Case Studies

The effectiveness, performance and advantage of the proposed RL-based wind farm control method are evaluated in this section with a dynamic wind farm simulator (called WFSim) designed in [36].

A. Case Study I

In this case study, we employ a flow field with the size being 4818.8m $\times$ 2700m. The simulation time step is 1s. Following the design of WFSim, the induction state α_i is set to be the modified thrust coefficient [36]. There are a total of 30 NREL 5MW turbines in this flow field. Fig. 4 illustrates the farm's layout, and the flow field (bird's-eye view) is simulated under the greedy strategy (i.e. $\alpha_i \equiv 2$, $\gamma_i \equiv 0$, $i = 1, 2, ..., 30$). The wind speed across the flow field is indicated by the color bar. Moreover, the wind direction along the x-axis is set to be the benchmark, i.e. $\Upsilon = 0°$, with the clockwise direction to be the positive direction. In the case study, we employ a stochastic wind profile with wind speeds ranging from 9m/s to 13m/s, and its wind direction follows $\Upsilon \in [-10°, 10°]$.

We employ Python (with PyCharm software) to program the automatic grouping, multi-agent reinforcement learning and multi-agent transfer learning methods developed in this paper. We use the Keras API for deep neural network implementations and the MATLAB engine in Python to link with WFSim. We denote the proposed control method as "RL with AG & MATL". The maximum and minimum allowable yaw offsets (with respect to the free-stream wind direction) are set to be $\gamma_{\max} = 30°$ and $\gamma_{\min} = -30°$, respectively, and the allowable range of α_i is set to be $[0, 2]$. The single-step restrictions on control signals are set to be $|\delta\alpha_i| < 0.05$ and $|\delta\gamma_i| < 0.1°$.

For AG, we set $w_\gamma(\cdot) \equiv 0$ and the threshold of $\mathcal{IF}$ to be 0.3. The wind change threshold to check potential AG result changes is $|\delta\Upsilon| = 10°$. The depth limit is $b_d = 3$ (the depth of root vertices is 0). Moreover, for the multi-agent RL structure, we have $\xi = 0.99$, $T = 512$, $\epsilon = 0.1$, and $\lambda = 0.95$ (a parameter in GAE). The actor DNNs for all agents have the same structure. They have two hidden layers with 64 and 32 ReLU neurons, respectively, while the critic DNNs have three hidden layers with neuron numbers adapted to group sizes. In addition to the greedy strategy and the proposed control method, we also employ a pure RL method that does not apply AG and MATL for comparison purposes. We denote it as "RL w/o AG & MATL" in the figures.

By carrying out AG for the 30-turbine wind farm, a unique grouping result is obtained with the simulation settings given above, as demonstrated in Fig. 5. Specifically, this figure provides a bird's-eye view of the wind farm & flow field simulated in this case study. It is generated based on the simulation results of WFSim with the proposed RL method. In Fig. 5, each black bar indicates a wind turbine in the farm, and color changes indicate variations of wind speeds across the flow field. This figure provides an intuitive illustration of wake effects and the wake steering ability of the proposed method. In addition, it also allows us to mark and highlight

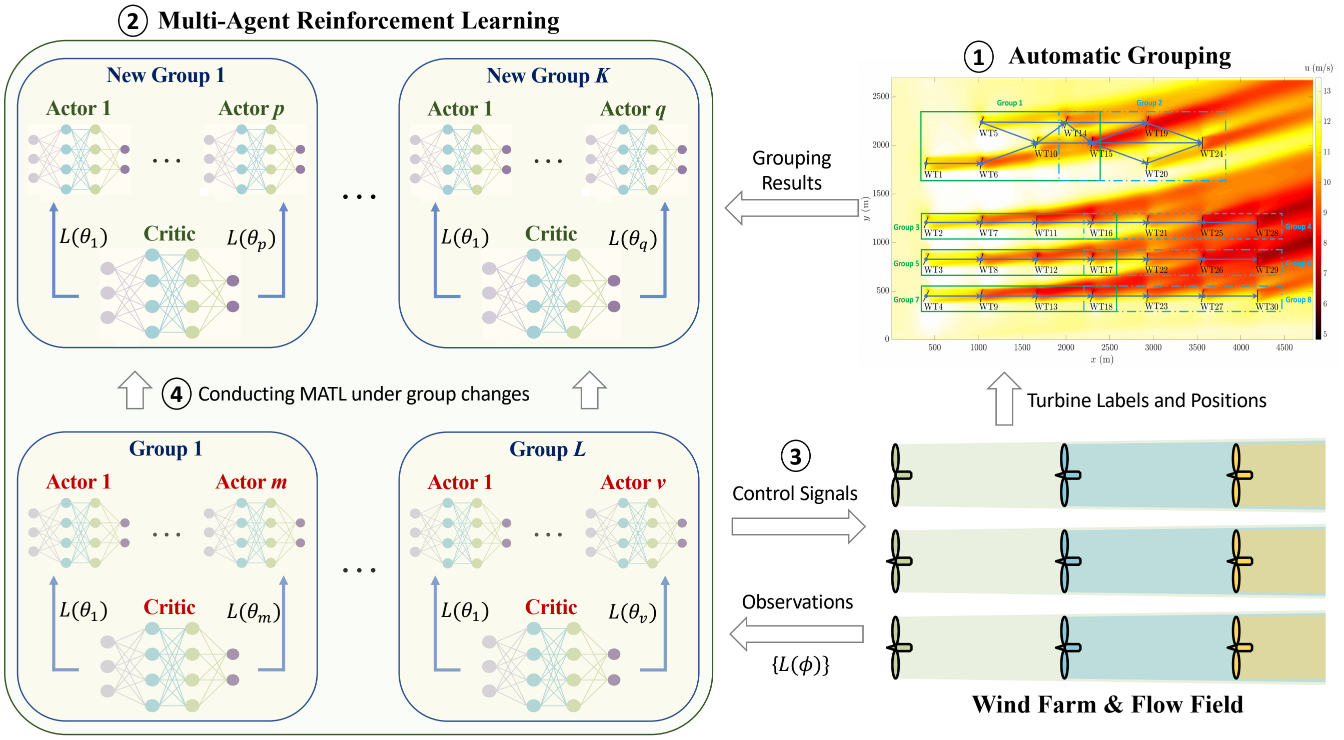

Fig. 3: Main structures of the proposed RL-based wind farm control method with AG and MATL.

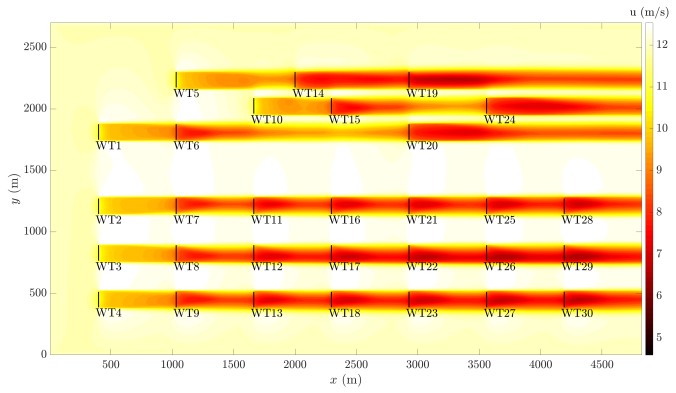

Fig. 4: The simulated flow field with a 30-turbine wind farm under the greedy strategy.

the results of AG (Algorithms 1 & 2). We discuss that in detail as follows.

Specifically, AG leads to four connected components (forming a large directed graph), and the vertices in these connected components (CC) are as follows.

CC1: $\{\mathcal{WT}_1, \mathcal{WT}_5, \mathcal{WT}_6, \mathcal{WT}_{10}, \mathcal{WT}_{14}, \mathcal{WT}_{15}, \mathcal{WT}_{19}, \mathcal{WT}_{20}, \mathcal{WT}_{24}\}$

CC2: $\{\mathcal{WT}_2, \mathcal{WT}_7, \mathcal{WT}_{11}, \mathcal{WT}_{16}, \mathcal{WT}_{21}, \mathcal{WT}_{25}, \mathcal{WT}_{28}\}$

CC3: $\{\mathcal{WT}_3, \mathcal{WT}_8, \mathcal{WT}_{12}, \mathcal{WT}_{17}, \mathcal{WT}_{22}, \mathcal{WT}_{26}, \mathcal{WT}_{29}\}$

CC4: $\{\mathcal{WT}_4, \mathcal{WT}_9, \mathcal{WT}_{13}, \mathcal{WT}_{18}, \mathcal{WT}_{23}, \mathcal{WT}_{27}, \mathcal{WT}_{30}\}$

With these connected components, AG divides the whole farm into eight turbine groups by carrying out RDFS with $b_d = 3$. We summarize the turbines in each group follows.

Group 1: $\{\mathcal{WT}_1, \mathcal{WT}_5, \mathcal{WT}_6, \mathcal{WT}_{10}, \mathcal{WT}_{14}, \mathcal{WT}_{15}\}$

Group 2: $\{\mathcal{WT}_{14}, \mathcal{WT}_{15}, \mathcal{WT}_{19}, \mathcal{WT}_{20}, \mathcal{WT}_{24}\}$

Group 3: $\{\mathcal{WT}_2, \mathcal{WT}_7, \mathcal{WT}_{11}, \mathcal{WT}_{16}\}$

Group 4: $\{\mathcal{WT}_{16}, \mathcal{WT}_{21}, \mathcal{WT}_{25}, \mathcal{WT}_{28}\}$

Group 5: $\{\mathcal{WT}_3, \mathcal{WT}_8, \mathcal{WT}_{12}, \mathcal{WT}_{17}\}$

Group 6: $\{\mathcal{WT}_{17}, \mathcal{WT}_{22}, \mathcal{WT}_{26}, \mathcal{WT}_{29}\}$

Group 7: $\{\mathcal{WT}_4, \mathcal{WT}_9, \mathcal{WT}_{13}, \mathcal{WT}_{18}\}$

Group 8: $\mathcal{WT}_{18}, \mathcal{WT}_{23}, \mathcal{WT}_{27}, \mathcal{WT}_{30}\}$

Based on that, RL can be executed on these small turbine groups. After the training episodes are finished, we test the performance of the proposed method by a long simulation with the total running steps to be 6000. Moreover, to evaluate the robustness against sudden wind direction changes, Υ is set to be

$$\Upsilon(t) = \begin{cases} 0^\circ & \text{for } 0 \le t \le 2000 \\ 5^\circ & \text{for } 2000 < t \le 4000 \\ 10^\circ & \text{for } 4000 < t \le 6000 \end{cases}$$

Simulation results of the flow field at $t = 2000$s and $t = 6000$s are given in Figs. 5 and 6, respectively. These figures indicate that the proposed method successfully achieves wake steering under stochastic wind speeds and time-varying wind directions.

The comparisons of farm-level power generations of different methods are provided in Fig. 7. This figure shows the changes of regulated power generations over time (normalized by the power production under the greedy strategy at $t = 0$s) of Greedy, “RL w/o AG & MATL” and “RL with AG & MATL”. One can see both the two RL-based wind farm control methods (the blue and red curves in Fig. 7) lead to clear farm-level power increases (22.23% and 20.29% on average, respectively) in comparison with Greedy (the yellow curve). Moreover, the proposed method has comparable performance to “RL w/o AG & MATL”, though there exists inevitable negligence of

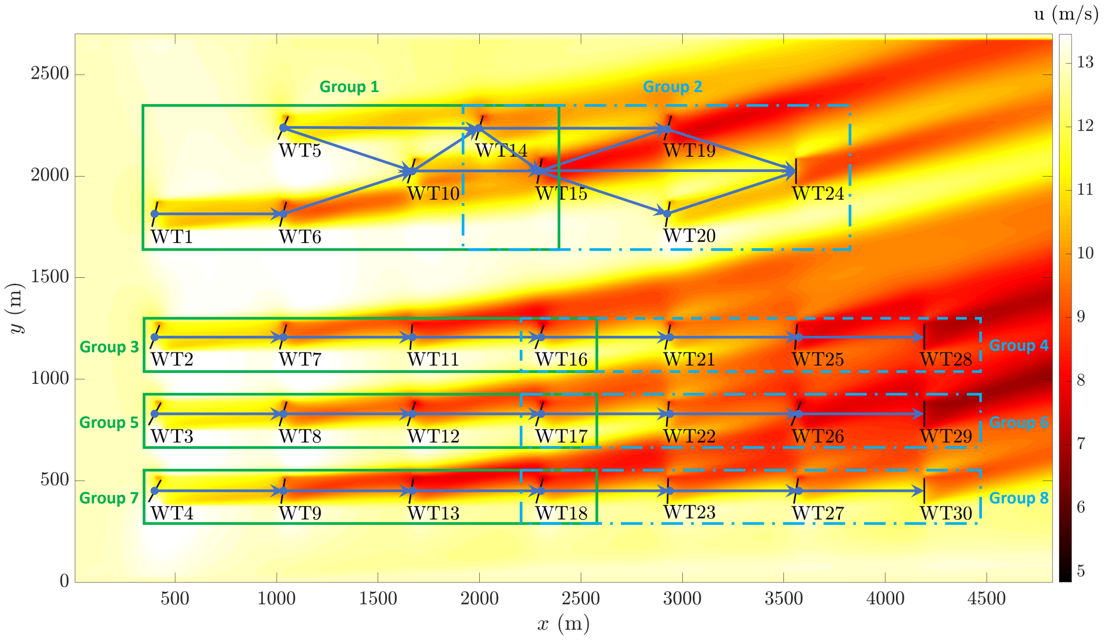

Fig. 5: Grouping and simulation results of the proposed wind farm control method.

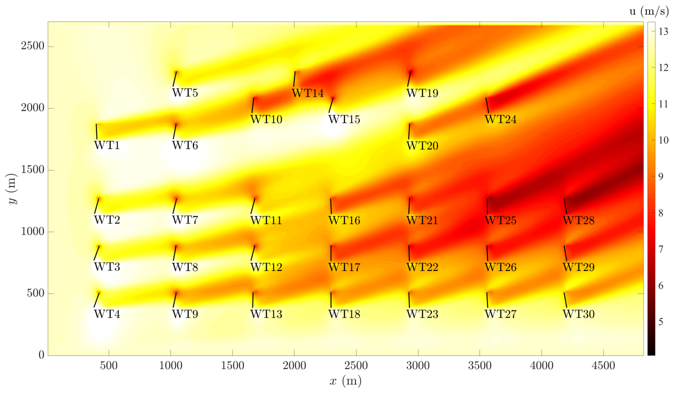

Fig. 6: The simulated flow field under the proposed method ($t = 6000$s, $\Upsilon = 10°$).

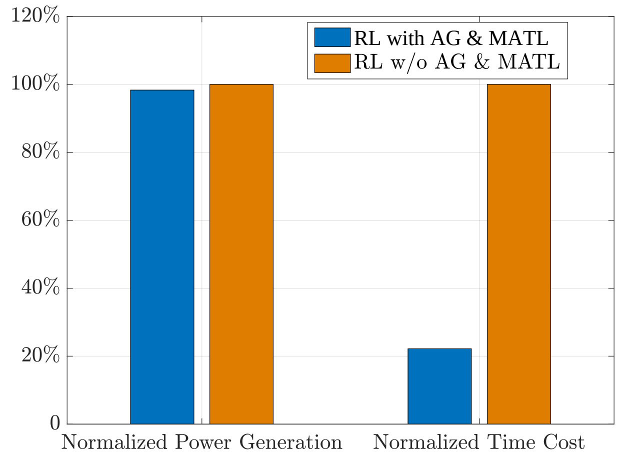

Fig. 8: Performance comparison of RL and RL-AG-MATL.

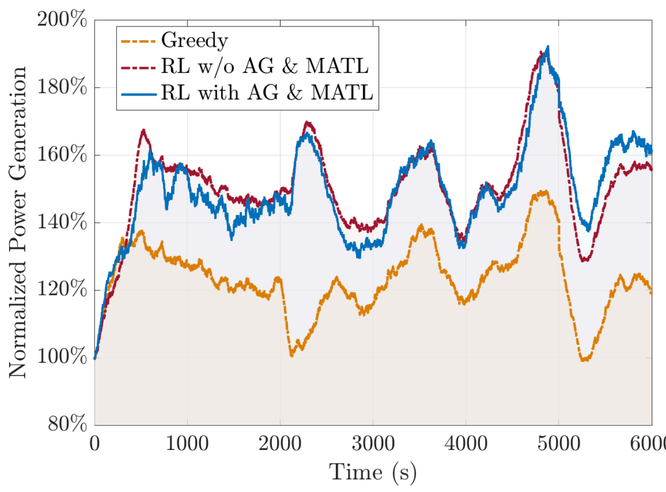

Fig. 7: Normalized power generation under different methods.

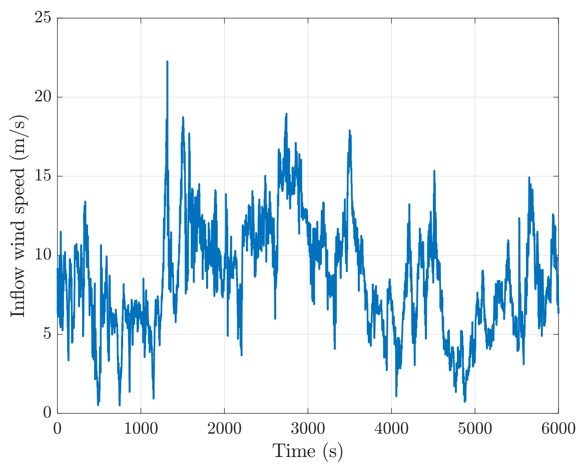

Fig. 9: Wind profile for Case Study II.

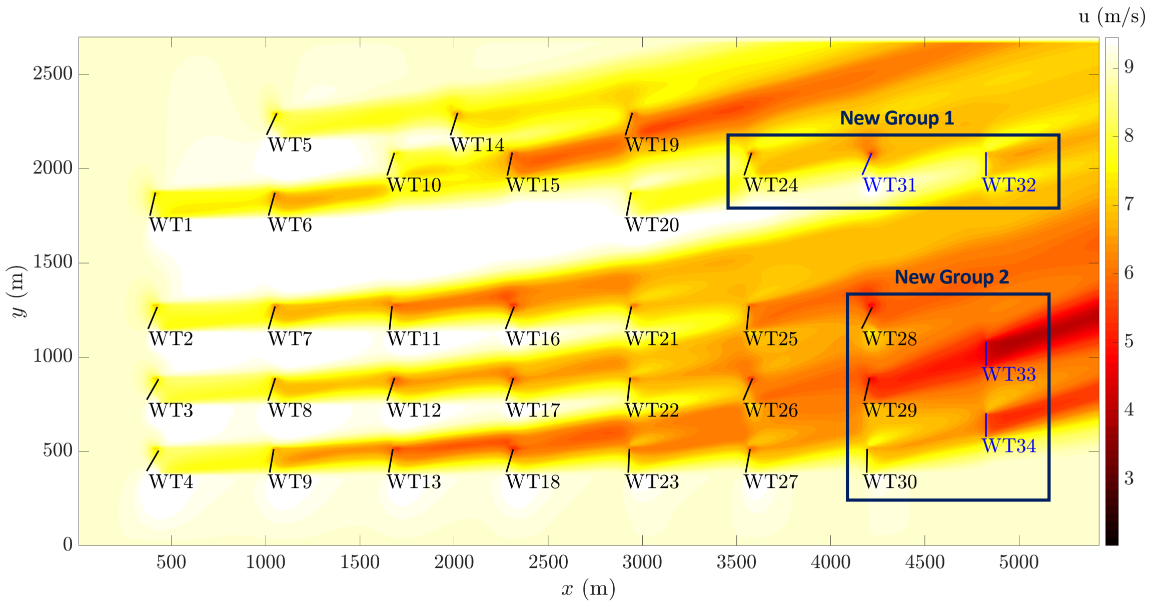

Fig. 10: Simulation results of the proposed wind farm control method with a 34-turbine wind farm.

aerodynamic interactions between groups. It even leads to better performance than pure RL for $5200 < t \leq 6000$. The performance of the two RL-based methods is further compared in Fig. 8. One can see that though applying AG in the proposed method results in a slight deficit (1.94%) in power increase percentage, it leads to significant computation load reduction (77.81%) in terms of the total training time (with Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GH, 96GB RAM).

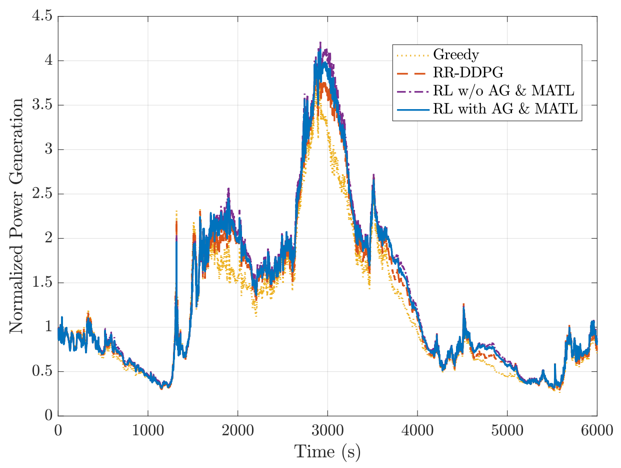

Fig. 11: Normalized power generation under different methods in Case Study 2.

B. Case Study II

This additional case study aims to test the scalability of the proposed method and compare it with other relevant results. We consider the situation that several new wind turbines ($\mathcal{WT}_{31}$-$\mathcal{WT}_{34}$) are installed in the wind farm, and their positions are indicated in Fig. 10. A wind profile based on the measurements of Anholt Offshore Wind Farm[1] is employed in this case study, as shown in Fig. 9 - one can see it has a large wind speed range with sharp changes. In addition to "Greedy", "RL w/o AG & MATL" and "RL with AG & MATL" methods, another approach developed in [31] is employed in simulations for performance comparison. This method also achieves data-driven wind farm control via deep RL. We denote it by "RR-DDPG" - it was based on the deep deterministic policy gradient (DDPG) algorithm [19] and employed a reward regularization (RR) module to handle wind speed changes. These features make it suitable to compare with the method proposed in the present paper.

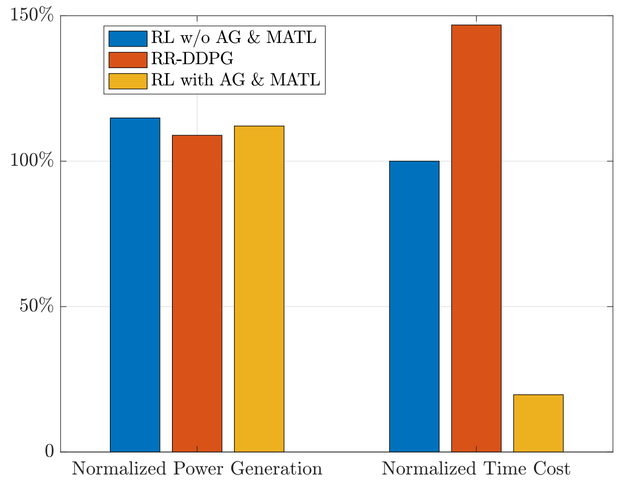

Fig. 12: Performance comparison of "RL w/o AG & MATL", "RR-DDPG", and "RL with AG & MATL".

Fig. 10 shows the simulation result of the flow field under the proposed method. An interesting observation is that AG

[1]https://orsted.com/en/our-business/offshore-wind

TABLE I: Comparison of different RL-based wind farm control methods.

Methods / Items	PR-DRL [28]	RR-DDPG [31]	KA-DDPG [24]	RL-AG-MATL (this work)
Data-driven & learning ability	Yes	Yes	Yes	Yes
Control target	Power tracking	Power maximization	Power maximization	Power maximization
Adaptability to wind changes	Yes	Yes	No	Yes
Reduced training costs	No	No	No	Yes

leads to two new turbine groups (as marked in Fig. 10) while all the other groups remain the same as in Case Study 1. This observation shows another advantage of the wind farm control method proposed in this paper - it has scalability when the size and layout of the wind farm change. In this case study, one only needs to apply RL to the New Groups 1 & 2. The control strategies of all the other turbines in the farm can be inherited from previous results - this avoids full retraining and significantly reduces training costs after size/layout changes.

The normalized power generations over time (normalized by the power production under the greedy strategy at $t = 0$s) under different control methods are depicted in Fig. 11. One can see the power generations change sharply under the wind profile in Fig. 9. Fig. 11 indicates that all three RL-based wind farm control methods have better farm-level power generations than the greedy strategy, showing the effectiveness and feasibility of applying RL to wind farm control tasks. The normalized power generation (w.r.t to Greedy) and computational time cost (w.r.t "RL w/o AG & MATL") under "RL w/o AG & MATL", "RR-DDPG", and "RL with AG & MATL" are given in Fig. 12 to compare their performance further. The proposed method leads to higher farm-level power generation than "RR-DDPG" (112.09% and 108.90%, respectively), and it has comparable performance to "RL w/o AG & MATL" (114.87%). As shown in the right-hand side of Fig. 12, a key feature of the proposed method is the significantly reduced training time. In this example, it only needs 19.72% of the training time of "RL w/o AG & MATL", while the time required by "RR-DDPG" is 7.44 times that of "RR with AG & MATL".

In addition, we summarize and compare the features of the proposed method and recently developed RL-based wind farm control approaches in Table I to provide more information. These methods include the preview-based robust reinforcement learning method (PR-DRL) proposed in Ref. [28], the RR-DDPG method proposed in [31] (which is employed in Case Study II), and the knowledge-assisted DDPG method (KA-DDPG) proposed in [24]. In summary, case studies and Table I show the performance and advantages of the proposed method. It can mitigate wake effects and improve farm-level power generation while significantly reducing the computational complexity associated with RL at the cost of mild performance degradation.

V. Conclusion

This paper developed a new reinforcement learning (RL)-based wind farm control method to improve the whole farm's power generation subject to strong aerodynamic interactions between turbines and time-varying environmental conditions. The proposed method had the ability to mitigate the high computational complexity associated with RL & deep neural network training via two novel designs: automatic grouping and transfer learning. The main RL structure was built upon a multi-agent actor-critic structure, allowing turbines to get control signals via local observations. Simulation results showed that our method was able to trade off control performance and computational loads. On the one hand, the resulting farm-level power generation under our method was higher than the conventional greedy strategy. On the other hand, compared to directly implementing RL across the entire wind farm, the proposed method could significantly reduce the training time at the cost of slight performance degradation, laying a foundation for RL's applications in the operational tasks of large wind farms. A limitation of the proposed method is that it was data-driven and did not make full use of physical information/models. Hybrid AI methods will be explored in the future to combine helpful physical information with deep RL algorithms and applied to wind farm control tasks. This can potentially improve the initialization and decision-making of RL, enhancing learning effectiveness and efficiency further. Another meaningful research direction is employing automatic grouping strategies in power tracking tasks, bringing flexibility and adaptability to ancillary services with wind farms.

References

[1] GWEC, "GWEC global wind report 2022," Tech. Rep., 2022.

[2] H. Dong, J. Xie, and X. Zhao, "Wind farm control technologies: From classical control to reinforcement learning," *Progress in Energy*, no. 4, 2022.

[3] P. A. Fleming, A. Ning, P. M. Gebraad, and K. Dykes, "Wind plant system engineering through optimization of layout and yaw control," *Wind Energy*, vol. 19, no. 2, pp. 329–344, 2016.

[4] F. Gonzalez-Longatt, P. Wall, and V. Terzija, "Wake effect in wind farm performance: Steady-state and dynamic behavior," *Renewable Energy*, vol. 39, no. 1, pp. 329–338, 2012.

[5] R. Shakoor, M. Y. Hassan, A. Raheem, and Y.-K. Wu, "Wake effect modeling: A review of wind farm layout optimization using jensen's model," *Renewable and Sustainable Energy Reviews*, vol. 58, pp. 1048–1059, 2016.

[6] P. M. O. Gebraad, F. Teeuwisse, J. Van Wingerden, P. A. Fleming, S. Ruben, J. Marden, and L. Pao, "Wind plant power optimization through yaw control using a parametric model for wake effects - a CFD simulation study," *Wind Energy*, vol. 19, no. 1, pp. 95–114, 2016.

[7] M. Vali, V. Petrović, S. Boersma, J.-W. van Wingerden, L. Y. Pao, and M. Kühn, "Adjoint-based model predictive control for optimal energy extraction in waked wind farms," *Control Engineering Practice*, vol. 84, pp. 48–62, 2019.

[8] Z. Dar, K. Kar, O. Sahni, and J. H. Chow, "Windfarm power optimization using yaw angle control," *IEEE Transactions on Sustainable Energy*, vol. 8, no. 1, pp. 104–116, 2016.

[9] M. F. Howland, S. K. Lele, and J. O. Dabiri, "Wind farm power optimization through wake steering," *Proceedings of the National Academy of Sciences*, vol. 116, no. 29, pp. 14 495–14 500, 2019.

[10] M. F. Howland, J. B. Quesada, J. J. P. Martinez, F. P. Larrañaga, N. Yadav, J. S. Chawla, V. Sivaram, and J. O. Dabiri, “Collective wind farm operation based on a predictive model increases utility-scale energy production,” *arXiv preprint arXiv:2202.06683*, 2022.

[11] N. Gionfra, G. Sandou, H. Siguerdidjane, D. Faille, and P. Loevenbruck, “Wind farm distributed pso-based control for constrained power generation maximization,” *Renewable energy*, vol. 133, pp. 103–117, 2019.

[12] J. Park and K. H. Law, “Bayesian ascent: A data-driven optimization scheme for real-time control with application to wind farm power maximization,” *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1655–1668, 2016.

[13] J. Park and K. H. Law, “Cooperative wind turbine control for maximizing wind farm power using sequential convex programming,” *Energy Conversion and Management*, vol. 101, pp. 295–316, 2015.

[14] M. A. Ahmad, M. R. Hao, R. M. T. R. Ismail, and A. N. K. Nasir, “Model-free wind farm control based on random search,” in *2016 IEEE international conference on automatic control and intelligent systems (I2CACIS)*. IEEE, 2016, pp. 131–134.

[15] X. Yin and X. Zhao, “Deep neural learning based distributed predictive control for offshore wind farm using high-fidelity les data,” *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3251–3261, 2020.

[16] K. Chen, J. Lin, Y. Qiu, F. Liu, and Y. Song, “Model predictive control for wind farm power tracking with deep learning-based reduced order modeling,” *IEEE Transactions on Industrial Informatics*, 2022.

[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[18] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, “Mastering the game of go without human knowledge,” *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” in *International Conference on Machine Learning*, 2016.

[20] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, “Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 413–14 423, 2020.

[21] Y. Yang, L. Juntao, and P. Lingling, “Multi-robot path planning based on a deep reinforcement learning dqn algorithm,” *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 177–183, 2020.

[22] H. Dong, X. Zhao, Q. Hu, H. Yang, and P. Qi, “Learning-based attitude tracking control with high-performance parameter estimation,” *IEEE Transactions on Aerospace and Electronic Systems*, 2021.

[23] Z. Xu, H. Geng, B. Chu, M. Qian, and N. Tan, “Model-free optimization scheme for efficiency improvement of wind farm using decentralized reinforcement learning,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 12 103–12 108, 2020.

[24] H. Zhao, J. Zhao, J. Qiu, G. Liang, and Z. Y. Dong, “Cooperative wind farm control with deep reinforcement learning and knowledge-assisted learning,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 6912–6921, 2020.

[25] P. Stanfel, K. Johnson, C. J. Bay, and J. King, “Proof-of-concept of a reinforcement learning framework for wind farm energy capture maximization in time-varying wind,” *Journal of Renewable and Sustainable Energy*, vol. 13, no. 4, p. 043305, 2021.

[26] H. Dong, J. Zhang, and X. Zhao, “Intelligent wind farm control via deep reinforcement learning and high-fidelity simulations,” *Applied Energy*, vol. 292, p. 116928, 2021.

[27] S. Vijayshankar, P. Stanfel, J. King, E. Spyrou, and K. Johnson, “Deep reinforcement learning for automatic generation control of wind farms,” in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 1796–1802.

[28] H. Dong and X. Zhao, “Wind-farm power tracking via preview-based robust reinforcement learning,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1706–1715, 2021.

[29] J. Xie, H. Dong, X. Zhao, and A. Karcanias, “Wind farm power generation control via double-network-based deep reinforcement learning,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2321–2330, 2021.

[30] H. Dong and X. Zhao, “Intelligent wind farm control via grouping-based reinforcement learning,” in *European Control Conference*, 2022.

[31] H. Dong and X. Zhao, “Composite experience replay-based deep reinforcement learning with application in wind farm control,” *IEEE Transactions on Control Systems Technology*, vol. 30, no. 3, pp. 1281–1295, 2021.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[34] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, “The surprising effectiveness of ppo in cooperative, multi-agent games,” *arXiv preprint arXiv:2103.01955*, 2021.

[35] B. Bollobás, *Modern graph theory*. Springer Science & Business Media, 2013, vol. 184.

[36] S. Boersma, B. Doekemeijer, M. Vali, J. Meyers, and J.-W. van Wingerden, “A control-oriented dynamic wind farm model: WFSim,” *Wind Energy Science*, vol. 3, no. 1, pp. 75–95, 2018.

[37] S. Boersma, B. Doekemeijer, S. Siniscalchi-Minna, and J. van Wingerden, “A constrained wind farm controller providing secondary frequency regulation: An les study,” *Renewable energy*, vol. 134, pp. 639–652, 2019.

[38] B. M. Doekemeijer, D. van der Hoek, and J.-W. van Wingerden, “Closed-loop model-based wind farm control using floris under time-varying inflow conditions,” *Renewable Energy*, vol. 156, pp. 719–730, 2020.

[39] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” *arXiv preprint arXiv:1506.02438*, 2015.

Hongyang Dong is an Assistant Professor in the School of Engineering, University of Warwick, Coventry, UK. He worked as a Research Fellow in Machine Learning and Intelligent Control at the University of Warwick from 2019 to 2022, before he became an assistant professor in November 2022. He obtained a PhD degree in Control Science and Engineering from Harbin Institute of Technology, Harbin, China, in 2018. His current research interest is control theories and machine learning methods with their applications in complex systems, including offshore renewable energy systems and autonomous systems.

Xiaowei Zhao received the Ph.D. degree in control theory from Imperial College London, London, U.K., in 2010. He was a Postdoctoral Researcher with the University of Oxford, Oxford, U.K., for three years before joining the University of Warwick, Coventry, U.K., in 2013. He is currently Professor of Control Engineering and an Engineering and Physical Sciences Research Council Fellow with the School of Engineering, University of Warwick. His main research interests include control theory and machine learning with applications in offshore renewable energy systems, smart grids, and autonomous systems.