Mean Field Multi Agent Reinforcement Learning for Active Wake Control

Ion Plămădeală

Supervisor(s): Mathijs de Weerdt, Grigory Neustroev

[1]EEMCS, Delft University of Technology, The Netherlands

Abstract

The wake effect which is turbulence behind a wind turbine created when it extracts energy negatively impacts the power output of the downstream turbines. Active Wake Control can mitigate this effect, by rotating some turbines away from the wind.

Previous research applied single agent reinforcement learning to apply Active Wake Control, showing good results for small-scale layouts, that don't scale for larger, practical wind farms.

To that extent, this study focuses on **the application of mean-field multi-agent reinforcement learning to Active Wake Control**, under constant wind conditions. This algorithm limits the computations to a limited set of neighbouring turbines, reducing their complexities. To build the answer to this question I will also study:

1. how to model the rewards to solve the lazy-agent problem, leveraging the nature of the Active Wake Control
2. how the view of the agent changes the results
3. how does it compare to a single-agent reinforcement learning algorithm, TD3

The experiments were done using the Floris Wake Simulator, with each turbine sharing the same agent, placed in tunnel layouts at real-life distances (6-7 rotor diameters), under constant wind conditions.

Results show that with the proper configuration of rewards and view space within wind tunnels, the mean-field algorithm finds near optimal configurations for Active Wake Control, within a small number of episodes. This shows a promising start for the application of mean-field multi-agent algorithms for the Active Wake Control problem, and provides insight into how to model the rewards, which might be applicable for the whole class of algorithms.

1 Introduction

The rapid expansion of wind farms worldwide necessitates the development of advanced control strategies to enhance their performance and address operational challenges. One of the problems that affect dense wind farms is **wake effect**, which is a zone behind a turbine where the wind speed is reduced. The wake effect impacts the downstream turbines, reducing the amount of energy that they can extract and increases the experienced stress.

A solution to reduce the wake effect is **Active Wake Control** (AWC). It involves actively changing the yaw of the turbine out of the wind. The yaw is the rotation of the turbine around the vertical axis, as can be seen in figure 1. While it reduces the energy extracted by the turbine that is moving away from the wind, it allows for more energy to be collected by the next turbines increasing the overall energy output of the whole farm. Research into the application of AWC [1] concludes that for full-scale commercial wind farms, yaw-based control shows improvements in both power generation (0.41% to 1.28%), and in the increased lifetime of the turbine (due to the reduced load factors on all the components) by 0.6-0.9%. For a practical farm, like Princess Amalia, rated for 125000 households [2], and at 1.28% improvement, would result in an increase of 1600 homes that can be supplied with electricity.

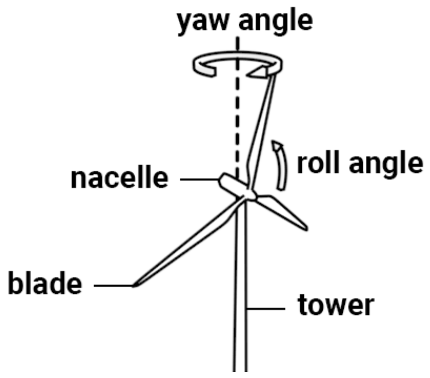

Figure 1: Turbine Nomenclature

The improvement is based on finding the best yaw for each turbine in the whole farm. The problem itself can be modeled as a Markov Decision Process (MDP). The MDP, a 4-tuple (S, A, P_a, R_a) corresponds to the problem as follows:

1. State - The wind direction, wind speed and the yaw of each turbine
2. Actions - for each turbine rotate left, right or stay in place
3. Transition Function - determines the resulting state for each turbine after a turn
4. Reward Function - The total power output in MWh of the resulting state

While an MDP can be solved with Dynamic Programming, the solution is too expensive for a practical farm, so we turn to the Reinforcement Learning (RL) class of algorithms.

Previous research [3] shows that single-agent reinforcement learning techniques are inadequate for a large scale farm, due to the combinatorial nature of the problem, and then offers as a possibility multiple-agent reinforcement learning (MARL). In this representation of the problem, each turbine would be assigned an agent, working together towards maximising the power output of the farm.

The field of MARL algorithms has grown a lot in the previous years. In order to properly describe and understand the algorithms, Oroojlooy and Hajinezhad (2023) [4] mentions some critical properties that define multi-agent algorithms: centralised/decentralised control, fully/partially observable environment, cooperative/competitive environment.

Their paper also classifies the algorithms into Independent Q-Learning (IQL) , Fully Observable Critics (FOC), Value Function Factorisation (VFF), Consensus and Learn to Communicate.

The categories above each solve some problems of the Multi Agent Reinforcement Learning. Mainly:

1. IQL: distributed the computation and responsibility of the single agent model, but does not scale with high dimensions, and does not consider the influence it has on the environment
2. FOC: helps with the non-stationary problem that is common to MARL
3. VFF: helps understanding which part of the reward comes from which agent, preventing lazy agents
4. Consensus and Learn To Communicate: helps with sharing only the relevant information.

Some of the problems listed above are not issues that come up with the AWC problem.

1. The interactions are localised, so we can limit the information of each agent to only the information from its neighbours, solving the communication issue and the high-dimensionality issue.
2. The turbines are stationary, and while they change with the direction of the wind, that is beyond the scope of the current paper.
3. Natural reward representation, the output power for each turbine.

From the advantages above, I came to the conclusion that an IQL Algorithm fits the requirements of the problem.

Mean Field [5] is a MARL (abbreviated in the future in the paper as MF MARL) algorithm that reduces the computation by choosing a limited amount of agents as neighbours with which to estimate the quality of their actions, ignoring the rest. This meaningfully reduces the complexity of the problem. At the same time, Blume (1993) [6] concludes that the global pair-wise interactions between all the agents are maintained, even when only local interactions are analysed. This comes as an added benefit in case the choice of neighbours is not expansive enough. It works as both Agent-Critic and IQL.

MF-MARL has been used in the past on problems that contain a lot of agents, with the closest to a AWC being a mixed cooperative-competitive battle game, that have two armies fighting in a grid world. (battle agent) The similar aspect with AWC is the presence of agents with a limited field-of-view that need to cooperate for a common goal. The number of agents used in the MF-MARL paper is 64 per team, which is similar to the number of turbines in the Amalia Farm.

With this in mind, I analyse the following research question: Can MF-MARL be applied to AWC?

To build the answer to this question I will study:

1. how to model the rewards to solve the lazy-agent problem, leveraging the nature of the AWC
2. how the view of the agent changes the results
3. how does it compare to a single-agent reinforcement learning algorithm, TD3

To answer this question, I will run some experiments, the methodology of which is described in section 2. My contribution to the base MF-MARL algorithm is presented in section 3. The exact RL runs are listed in section 4, with their results The ethical implications of the research is presented in I interpret the results of my runs in the Finally, I conclude the paper in

2 Methodology

In order to understand how well the algorithm does, I will run multiple experiments each with progressively more wind turbines. Below I present the adaptation of MF MARL to AWC, and the experiment settings.

2.1 Mean-Field Q-Learning

Categorising using the framework proposed by Oroojlooy (2023), MF-MARL is a centralised, partial-information, cooperative algorithm. It is made out of two Deep Q-value estimators: evaluator and target .

The Q-value estimator is a Deep Neural Network, presented in figure 2. Being a centralised algorithm, the target/evaluator q-estimators are shared for the whole farm.

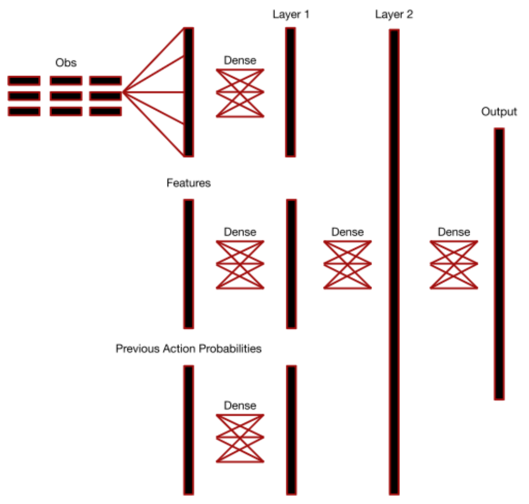

Figure 2: The architecture of the Q-value estimator

The input features for each turbine is the wind speed and direction as measured at the turbine, and its current yaw angle.

The observation space of each turbine, represented in figure 3, consists of the features of its neighbours. The size is parameterised by a radius tuple, r_x, r_y that represents the horizontal and vertical size of the view space. The view itself is a rectangular field of size $(1+radius_x \cdot 2) \times (1+radius_y \cdot 2)$. It can be represented as cells mapping to a $750m \times 750m$ square plot from the wind farm, and represents the features of a turbine in that location.

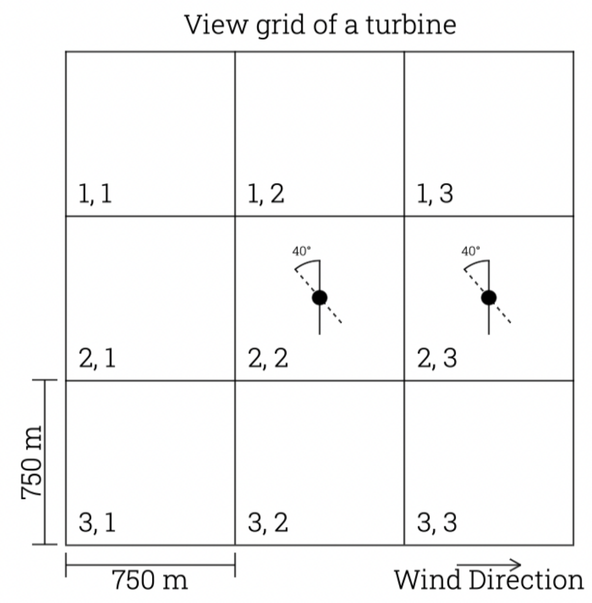

Figure 3: Grid View of a turbine

As an example, for figure 3, the turbine being analysed is the one in cell (2, 2), $t_{2,2}$. Assuming that feature vector for $t_{i,j}$ is $\vec{f_{i,j}}$, the final observation matrix would be: $\begin{bmatrix} \vec{0} & \vec{0} & \vec{0} \\ \vec{0} & \vec{f_{2,2}} & \vec{f_{2,3}} \\ \vec{0} & \vec{0} & \vec{0} \end{bmatrix}$

The action space is discrete , where each action is represented as -1 (turn full speed to the left), 0 (stay in place), 1 (turn full speed to the right). This corresponds to the output of the Q-estimator, where for each action the model assigns a q-value.

2.2 Experimental Setup

The experiments themselves will be run using a wake simulator, Floris [7]. Each experiment is composed of 3000 or 4000 episodes, based on the size of the wind farm, and features epsilon-greedy exploration of the action space. As a refresher, the epsilon greedy exploration, parameterised by a number between 0 and 1 representing the probability of choosing an action at random or letting the agent pick their best move. For my experiments, the epsilon parameter is linked to the episode number, and follows a linear-decay pattern:

The length of each episode is 150 steps, starting with all turbines facing the wind. The wind and speed direction is constant, set to come from the left, and the maximum yaw for each turbine is in the range -40° to +40° from the initial position. The learning rate is 0.0001, with a Polyac update

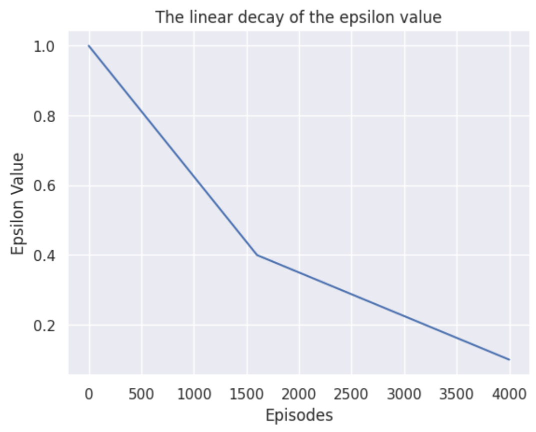

Figure 4: Decay of the epsilon value according to time

tau value of 0.05. For the performance metric, I have used the mean power output of the whole farm over one episode.

2.3 Wind Farm Layouts

The experiments are centred around 2 layouts. The first one, depicted in figure 5, the wind tunnel has 3 turbines placed in a row, at a distance of 750m from one another. This scenario is useful as it can be run efficiently and can inform about the feasibility of a model in a maximally adversarial situation.

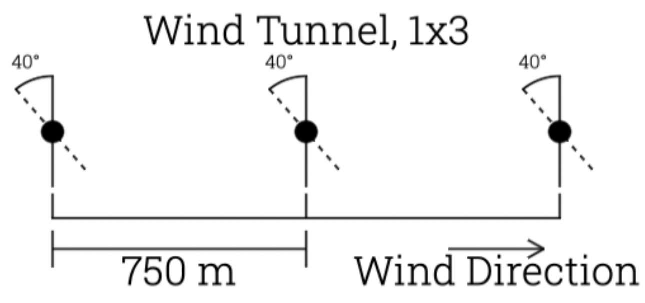

Figure 5: Layout for the experiment using 3 turbines.

For the second layout, I have a 4x4 grid of turbines. This should test how the inclusion of irrelevant turbines, that are not downstream from the wind influences the learning properties of the algorithm. This builds toward a real world farm layout where the wind changes its direction (changing which can be considered downstream turbines) and the layout is packed, requiring the algorithm to learn the difference between relevant and irrelevant observations.

The rewards are defined in section 3.2.

3 Changes to the base MF-MARL algorithm

I have made the following changes to the original algorithm that adapted it to the AWC problem:

Q-Estimator Architecture The original MF-MARL utilises a depth-2 convolutional layer on the observational input. Since the input space is small, and the position of a turbine does influence the actions that it takes, I decided to skip this layer for my architecture.

3.1 Training process

Constant Temperature I changed the training process to no longer change the softmax temperature following a linear decay path, but it is constant through the whole training process. The exploration provided by the changing temperature is replaced with Epsilon-Greedy exploration.

Introduced Epsilon-Greedy Exploration strategy As discussed in chapter 2, as part of the training process, I have introduced an epsilon greedy exploration strategy. This nudges the algorithm towards an optimal path, since without it, the algorithm is stuck at the greedy baseline (having the turbines just face the wind), even with variable temperature.

3.2 Wind Downstream Rewards

The reward that I have used is the sum of the differences between the output energy of the current step and the output energy of the previous step, for all the turbines present in the reward space of the turbine. It is equivalent to the observational space, but is limited to only turbines from upstream and downstream the wind.

Again with the example from chapter 2.1, assuming that feature vector for $t_{i,j}$ at time step k is $\vec{f_{i,j,k}}$, with output (in MWh) $o_{i,j,k}$ the observation matrix for 2,2 would be:

$$obs = \begin{bmatrix} \vec{0} & \vec{0} & \vec{0} \\ \vec{0} & \vec{f_{2,2,k}} & \vec{f_{2,3,k}} \\ \vec{0} & \vec{0} & \vec{0} \end{bmatrix}$$

To obtain the difference in power between steps,

$$\Delta_{i,j,k} = o_{i,j,k} - o_{i,j,k-1}$$

. Then, the reward view $v_{i,j}$ for the turbine:

$$v_{2,2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \Delta_{2,2,k} & \Delta_{2,3,k} \\ 0 & 0 & 0 \end{bmatrix}$$

With the final reward for the turbine defined as:

$$r_{i,j,k} = \sum_{v_{i,j}} \Delta$$

$$r_{2,2,k} = \Delta_{2,2,k} + \Delta_{2,3,k}$$

Since the view of the rewards is independent from the observational view, it can be adjusted for each step, and be adjusted for the wind direction. The implications of the change can be seen throughout the experiments.

4 Experimental Setup and Results

4.1 Wind Tunnel, 1x3

To prove that the algorithm can find a solution to the problem, and can learn from the environment, I start with a Wind Tunnel, in the configuration discussed in chapter 2. The observed space for the agent is set to $r_x = 1, r_y = 1$. The running time for each experiment was the experiment was approximately 2h, with on average 1.8 seconds per step.

Reward: Total power output For the reward, each turbine gets the total power output of the wind farm: $r_i = \sum_i^3 o_i$ This reward is inspired from the transition from a single agent reinforcement learning to multi-agent reinforcement learning.

Reward: Global Delta Sum For this experiment, the reward given to each turbine is the sum of the differences between the output power between last step and current step: Using the following notation: t_i - ith turbine, $o_{i,k}$ - output of ith turbine at step k, $\Delta_{i,k}$ - delta for turbine i between episodes, $r_{i,k}$ - reward used for turbine i:

$$\Delta_{i,k} = o_{i,k} - o_{i,k-1}$$

$$r_{i,k} = \sum_{j}^{3} \Delta j, k$$

Reward: Limited View Delta Sum For this experiment, the reward given to each turbine is only the sum that can be gathered from the observation view. It is described in more detail in section 3.2.

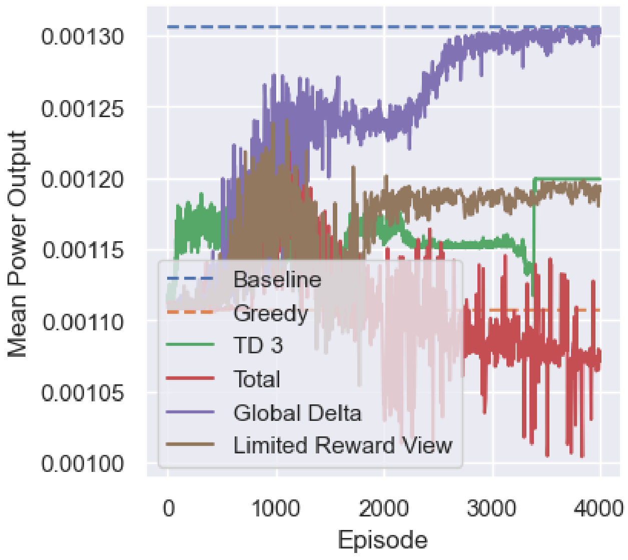

Figure 6: Mean power output per episode 1x3

Field View: Too Small For this experiment, the reward is the global delta sum, but the observation field is $r_x = 0, r_y = 0$ (view of size 1), only including self.

4.2 Experiment with parallel wind tunnels, 4x4

To understand the effect of irrelevant information in the view of the turbine, coming from parallel turbines from the direction of the wind, I have set up this experiment, as detailed in section 2. The experiments took an average of 9 hours, with about 11 seconds per episode. For these experiments I reduced the number of episodes to 3000, due to time constraints.

Reward: Global Delta As described in the 1x3 Tunnel.

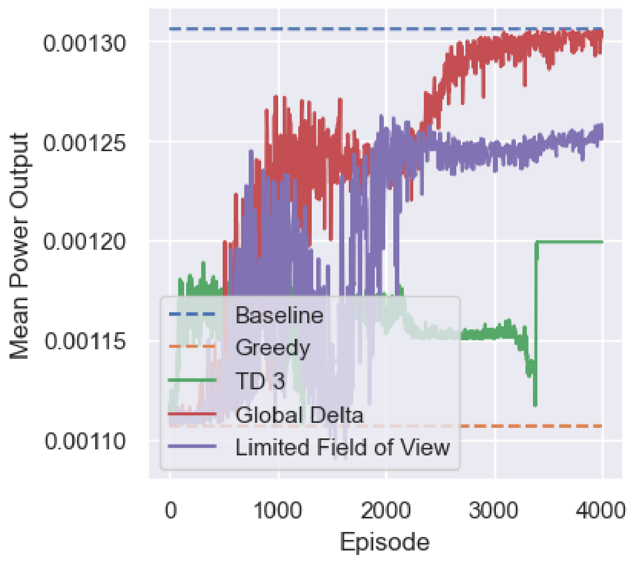

Figure 7: Mean power output per episode 1x3

Reward: Normal X/Normal Y Observation View, Reward View : $r_x = 1, r_y = 1$. This applies the basic algorithm, but adds confusing reward information from parallel turbines ($r_y = 1$)

Reward: Large X/Normal Y Observation View: $r_x = 1, r_y = 1$. Reward View: $r_x = 2, r_y = 0$. Only the relevant Rewards are applied, the observation space is unchanged.

Reward: Large X/Reduced Y Observation View: $r_x = 1, r_y = 0$. Reward View: $r_x = 2, r_y = 0$. Reducing the Observation View to only the relevant turbines, with only the relevant rewards.

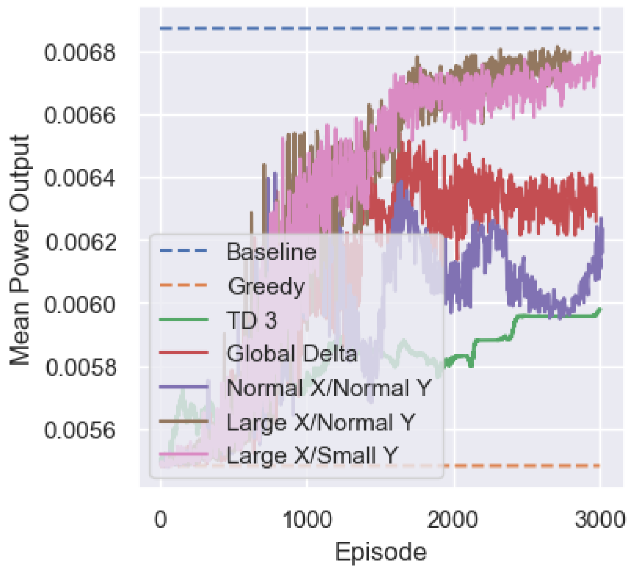

Figure 8: Mean power output per episode, 4x4

5 Responsible Research

The resources used in this paper, Floris, TD3 and the mean field implementation were part of the open domain, as published research. At the same time, the code and configurations used for this experiment are public on GitHub , providing an easy way to reproduce and verify the results listed in the paper. As such, all the components follow the FAIR [8] principles. Ethically, the project is targeting wind turbines, which does not lead to any bias or societal concern.

6 Discussion

The experiments listed in section 4 clearly show that at least for wind tunnels (turbines placed in a row for maximal negative impact of the greedy strategy), with constant wind direction MF-MARL can learn the optimal yaws for Active Wake Control, under the proper configuration.

To understand how to build the proper configuration, I experimented with different choices of rewards and sizes of the observation view.

6.1 Choice of reward

The choice of rewards yields the biggest changes in the algorithm, since it directly shapes what the algorithm learns about the quality of each action.

Power Output Giving the algorithm the power output does not work, for reasons that could be researched in the future. The trend, as seen in figure 7 is that the output decreases below the greedy (do nothing) result.

Global Delta Sum Giving the algorithm the global delta sum works well for a single wind tunnel, since the reward contains only relevant information, but confuses the algorithm on the larger layout (4x4), due to lazy agent problem, and sometimes has positive rewards for bad actions. It represents an improvement over TD3 (which also faces the lazy agent problem) from the distributed nature of the algorithm.

Delta Sum, but not enough information From the 1x3 experiment with a limited reward view, the mean power output trends towards the same result obtained by the TD3 algorithm. While an improvement over doing nothing, it is still below what could be achieved. This is an interesting result that shoes that if the interaction space is too small, it is not enough to simulate the global interactions.

Delta Sum, not enough and confusing information As expected, the combination of the two factors lead to an worse outcome, as it the case for the 4x4 Normal X/Normal Y experiment. In that experiment, the turbine has both a limited view and it includes irrelevant information. While the algorithm eventually reaches the level of global delta view, across multiple experiments it takes a more episodes to achieve that performance.

Delta Sum with proper, but limited information Once the right configuration of reward has been found, as it the case with the 4x4 matrix Large X experiments ($r_x = 2$), the algorithm converges to near-optimal results. Since some turbines do not have access to the full reward of the turbines they impact (for example the leading turbine does not know about the existence of the 4th turbine), it shows that the algorithm scales even with partial information.

6.2 Choice of Observation View

Too narrow a field view. In the last experiment 1x3 wind tunnel experiment, I show that the field of view does indeed play a role. In that experiment, I let the turbine only see it's own features in the observation field. Since it cannot understand whether it is first, in the middle or last, it cannot learn optimally. Since the algorithm is centralised, it cannot differentiate enough on rotation and wind speed alone. Still, they are enough to find a decent solution, that is above TD3 and greedy.

Too big of an observation view Most of the experiments run were done on a field of 1x1, and since the algorithm found near-optimal results, this is enough for the agent to understand whether it has a turbine in front, or one behind.

Reduce to only relevant information I have run the 4x4 Large X experiments with a reduced observation view (Small Y, $r_y = 0$) and a normal one (Normal Y, $r_y = 1$). This shows that the algorithm does not benefit at all from having the irrelevant information taken out, as it does not improve the performance of the algorithm. This answers the research sub-question, proving that the algorithm is capable of filtering out irrelevant observation information.

7 Conclusions and Future Work

This paper tries to understand how well can Mean Field Multi-Agent Reinforcement Learning to the problem of Active Wake Control, and whether this is an improvement over the greedy solution of facing the wind, and over single agent reinforcement learning solutions, for wind farm layout consisting of a wind tunnel and parallel wind tunnels. After lots of experiments, I found that with the proper choice of rewards, and the proper choice of the observation space, limited to the downstream and upstream turbines and modifications to the training process, it does indeed find near-optimal results for a limited number of configurations. Compared to the single-agent version of the algorithm, which suffers from exploration problems and lazy agents, it does perform better.

For future improvements of the algorithm, it can be tested on a real world layout, or non-parallel wind tunnels.

At the same time, the algorithm has only been tested on a constant wind direction, which is not reflective of the actual wind. This could also be coupled with creating an input space that is wind dependent, giving the agent only information about the downstream turbines. It could also be mixed in with an Value Function Factorisation approach, to create a function that can learn the value of the rewards for each turbine in a wind-dependent manner. In the case the performance of MF-Q is reduced when the wind direction changes due to the environment being no longer static, the Mean Field Actor Critic approach can be tried.

References

1. Kanev, S., Savenije, F. & Engels, W. Active wake control: An approach to optimize the lifetime operation of wind farms. *Wind Energy* **21,** 488–501. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.2173. https://onlinelibrary.wiley.com/doi/abs/10.1002/we.2173 (2018).
2. *MS Windows NT Kernel Description* https://www.power-technology.com/projects/princess-amalia/. Accessed: 2023-06-25. Dec. 2009.
3. Neustroev, G., Andringa, S. P. E., Verzijlbergh, R. A. & De Weerdt, M. M. *Deep Reinforcement Learning for Active Wake Control* in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (International Foundation for Autonomous Agents and Multiagent Systems, Virtual Event, New Zealand, 2022), 944–953. ISBN: 9781450392136.
4. Oroojlooy, A. & Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence* **53,** 13677–13722. ISSN: 1573-7497. https://doi.org/10.1007/s10489-022-04105-y (June 2023).
5. Yang, Y. *et al. Mean field multi-agent reinforcement learning* in *International conference on machine learning* (2018), 5571–5580.
6. Blume, L. E. The statistical mechanics of strategic interaction. *Games and economic behavior* **5,** 387–424 (1993).
7. *Floris* https://github.com/NREL/floris. Accessed: 2023-06-25.
8. *FAIR Principles* https://www.go-fair.org/fair-principles/. Accessed: 2023-06-25.