Applying QMIX to Active Wake Control

Multi-Agent Reinforcement Learning

Jasper van Selm

Supervisor(s): Mathijs de Weerdt[1], Greg Neustroev[1]

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

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Jasper van Selm
Final project course: CSE3000 Research Project
Thesis committee: Mathijs de Weerdt, Greg Neustroev, Przemyslaw Pawelczak

Abstract

When multiple wind turbines are positioned close to one another, such as in a wind farm, wind turbines located downwind of other turbines are not 100% efficient due to wakes, negatively affecting the total power output of the wind farm. A way to mitigate the loss of power is to steer the wake away from the next turbine, which lowers the current turbine's power output but increases the turbine's total power output. As the number of wind turbines increases, how complex it is to calculate the optimal steering increases exponentially. Reinforcement learning techniques have been a promising candidate to solve this problem. However, single-agent techniques are still very computationally expensive when the number of turbines is the same as an average wind farm. Therefore, this paper aims to see how the QMIX algorithm, a multi-agent reinforcement learning technique, can be efficiently applied to the problem of active wake control. QMIX will be compared to the FLORIS model and a single agent deep reinforcement learning technique TD3 to see if it achieves a higher average reward and converges faster. Finally, QMIX is tested on a larger wind farm to see if it achieves any results in a reasonable amount of time, showing that multi-agent reinforcement learning techniques are more suitable to the problem. This paper shows QMIX has the potential to outperform TD3; although FLORIS is better for smaller wind farms but more research has to be done in applying QMIX to the problem.

1 Introduction

1.1 Active Wake Control

When a wind turbine extracts energy from the wind, it creates a wake behind it, which is an area of high turbulence and low wind speeds. Wind farm engineers place turbines close together due to external constraints, so when this wake reaches another turbine, it negatively impacts the power output of that turbine [1]. By turning the turbines, a controller determines the direction of the wakes, and this process is known as active wake control in wind farms [2]. Wake-induced losses can be extremely high and range from 10-24%, depending on the layout of the turbines [3]. With the number of wind parks and their sizes increasing [4], the total wake-induced loss is increasing along with it. Therefore the problem of active wake control is becoming ever more critical.

More on the modelling side, research has been done into wake aerodynamics, such as how the wake decays downstream, to estimate the effect on downstream turbines in [1]. Researchers for wind farm control have created multiple strategies, and they differ by having different characteristics, such as being closed-loop controllers, the type of model they use, and the measurements they incorporate in the model. Different types of closed-loop control strategies include: optimization-based closed-loop control, where a controller is fed wind farm measurements and optimal control inputs are output; linear dynamic closed-loop control, where a dynamic controller is designed using linear models; and an observer who can estimate the entire state and use that for control. The fidelity of the model used also impacts the model as a higher fidelity means the model is more computationally expensive, making them less feasible for online control [5]. A possible way to circumvent this is to use reinforcement learning, as this discards the wake model and learns an optimal control policy for the wind farm.

1.2 Reinforcement Learning

Reinforcement learning (RL) is particularly well suited to the problem as it involves an agent transitioning from one state to the next, which is relevant to the dynamic nature of active wake control. Additionally, RL can work with continuous tasks where conditions change, which is especially useful for active wake control. Some research has been done in using reinforcement learning for active wake control in [2; 6; 7]. The work done in these papers shows promise for reinforcement learning to be used by controllers for the problem. However, more research can be done to improve it by looking at action encoding, scalability for large wind farms, and additional sensor data. Methods used in these papers are single-agent reinforcement learning techniques which suffer from the combinatorial explosion of adding more wind turbines as the action space grows exponentially with each added turbine. A way to mitigate this is to use multi-agent reinforcement learning, where each turbine is an agent which learns the best policy for that turbine and cooperates with other agents to achieve the highest overall power output.

Multiple multi-agent reinforcement learning techniques (MARL) exist; some are more suited to the problem than others. In [8], four multi-agent algorithms suit the characteristics of this problem, which include: Team-Q [9], Distributed-Q [10], QMIX [11] and MADDPG [12]. Both Team-Q and Distributed-Q are coordination-free methods [8]. Since each turbine heavily depends on the previous turbine's yaw, this method is only somewhat applicable. Comparing these, QMIX and MADDPG are a better fit for the problem as they have a centralized critic, which can learn the best policies more efficiently. For MADDPG, researchers have done some work by applying it to the active wake control problem, as seen in a paper titled: "Decentralized yaw optimization for maximizing wind farm production based on deep reinforcement learning" [13]. Since there is no existing work on using QMIX for active wake control, the decision was to explore the QMIX algorithm as it is a value-based algorithm that might work better with this problem than MADDPG, a policy-based algorithm.

2 Background

Before being able to answer the research question: *"How can the QMIX algorithm be efficiently applied to the problem of active wake control in wind farms?"*, a few topics and tools used in the paper should be introduced. These include an explanation of the inner workings of the QMIX algorithm; the OpenAI gym environment [14], which is utilised often in reinforcement learning research; and a background of FLORIS [15] and TD3 [16] which the paper will compare to QMIX.

2.1 QMIX

QMIX (Monotonic Value Function Factorisation) is based on the centralised training with decentralised execution paradigm and learns a joint action-value function. It is a model-free, value-based, off-policy algorithm that, in terms of technique, lies between independent Q-learning and counterfactual multi-agent policy gradients. Value decomposition networks (VDN) were the basis of QMIX, which is a non-linear extension. The difference is that QMIX does not do a complete factorisation like in VDN to extract decentralised policies.

To factorise the joint action-value function Q_{tot} into individual ones Q_a, the global argmax performed on Q_{tot} should give the same result as the set of individual argmax operations on each Q_a. QMIX does factorisation by representing the joint action-value function as a monotonic function using a constraint on the relationship between Q_{tot} and Q_a such that $\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A$.

The algorithm architecture consists of agent networks, a mixing network and a hyper-network. Agent networks represent the agent's individual value functions which are then combined into a centralised action-value function using a mixing network. An independent hyper-network generates each weight for the mixing network. This network takes the global state as input and outputs the weight of one layer. The algorithm combines the functions non-linearly to ensure consistency in the centralised and decentralised policies. Each agent can choose actions in a decentralised matter by calculating their value functions and using the mixing network to find the joint value function, after which the action with the highest value can be chosen [11]. The structure of the mixing network (a), overall QMIX architecture (b), and the agent network structure (c) are found in Figure 1. A detailed explanation of the implementation is found under section 4.1.

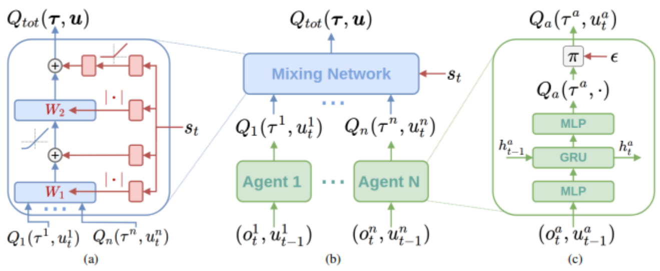
$Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u})$
W_2
$|\cdot|$
s_t
W_1
$|\cdot|$
$Q_1(\tau^1, u_t^1)$ $Q_n(\tau^n, u_t^n)$
(a)
$Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u})$
Mixing Network
s_t
$Q_1(\tau^1, u_t^1)$ $Q_n(\tau^n, u_t^n)$
Agent 1
Agent N
(o_t^1, u_{t-1}^1) (o_t^n, u_{t-1}^n)
(b)
$Q_a(\tau^a, u_t^a)$
π
ϵ
$Q_a(\tau^a, \cdot)$
MLP
h_{t-1}^a
GRU
h_t^a
MLP
(o_t^a, u_{t-1}^a)
(c)

Figure 1: QMIX architecture showing agent and mixing network structure [11]

2.2 OpenAI Gym Environment

The wind farm simulator used in [2] is based on the FLORIS (Flow Redirection and Induction in Steady-State) [15] model and was implemented as an OpenAI gym environment to make it easier for other RL researchers to use. An open-source library mainly used to develop RL algorithms, OpenAI gym has much helpful functionality for developing RL techniques. A gym environment has rewards, action spaces, and observation spaces, which a program can retrieve with simple function calls. Additionally, to observe the state and get the rewards, the only thing that a program must do is pass the actions to the environment. Therefore, the environment is a black box and can be used with just a few simple function calls, simplifying the RL algorithm creation process.

2.3 FLORIS

The FLORIS model includes an optimizer that works as follows: wake models and a set of atmospheric conditions are provided by the user, after which the steady-state wake locations and the wind flows throughout the farm are predicted. With this information, the optimizer creates a function that calculates the farm's total power output using the wind turbines' yaws. The optimizer then maximizes this function, which works well for a few turbines, but it soon becomes too complex when adding more turbines [2]. A few assumptions are made in the FLORIS model to reduce the computational complexity. One of these assumptions is that FLORIS assumes the wake flow is axisymmetric. In essence, this allows for a two-dimensional representation of the wake which is simplified compared to three-dimensional models. Another assumption is that the FLORIS model assumes the velocity deficit in the wake follows a Gaussian distribution. These assumptions make it less accurate and might not give the optimal solution.

2.4 TD3

Twin Delayed Deep Deterministic Policy Gradient (TD3) [16] is based on Deep Deterministic Policy Gradient (DDPG) [17] and aims to make the learning process more stable. TD3 is an actor-critic method, a class of algorithms often used in reinforcement learning consisting of two components: the actor and the critic. The actor is responsible for taking actions and learning an optimal policy, while the critic evaluates the value of the actions taken by the actor. Feedback is then given to the actor by the critic, and depending on whether the feedback is positive or negative, the actor's policy is updated, and this cycle repeats until convergence. TD3 makes the learning process more stable than DDPG by adding a second critic and updating the policy less frequently. Like DDPG, TD3 is a single-agent reinforcement learning technique with only one actor carrying out the policy.

3 Applying QMIX Efficiently

This paper investigates whether wind farms can efficiently apply the QMIX algorithm to solve the active wake control problem. The algorithm will be applied by using a wind farm simulator and comparing the performance of QMIX to FLORIS and a single agent RL technique TD3 [16]. Additionally, how fast QMIX learns will be compared by looking at the convergence speed for each technique. Since FLORIS and TD3 are not able to get any good results in a reasonable amount of time for large wind farms, QMIX will be tested on larger wind farms to see if it can get any results being a multi-agent reinforcement learning technique. By investigating this, whether QMIX is an excellent approach to solving the active wake control problem in wind farms should be answered.

In order to find out if the QMIX algorithm can be efficiently applied to the problem of active wake control, it is vital to look at what is meant by something being applied efficiently. In the case of active wake control for wind farms, an algorithm performs well by generating more energy on average, converging on an optimal solution in a period of time that the owner of a wind farm would be willing to pay for, and can work with large complex wind farms which are found in reality. The QMIX algorithm will be compared to TD3 and FLORIS to evaluate these metrics. In this section, an explanation of performance, convergence and complexity in terms of the problem of active wake control will be given.

3.1 Performance

The performance of the QMIX algorithm can be measured in different ways when the overall goal is to produce the most energy in the wind farm. A logical way to measure this would be to take the average power output of the entire wind farm over a certain period, as the individual turbine outputs are not important in the overall context. It is essential to have a baseline to compare the performance with; this is to have all turbines pointed in the direction from where the wind is coming. If the algorithm performs worse than this, there is no point in applying it because the wind farm achieves a higher power output without doing anything. Since QMIX and TD3 need to be trained, the performance can only be evaluated after training, and no training should occur during the evaluation. In this paper, therefore, the performance of an algorithm for the active wake control problem is to compare the average combined power output of all wind turbines over a specified time.

3.2 Convergence

In reinforcement learning, convergence is the process of an algorithm gradually improving its performance until reaching an optimal or near-optimal policy. Convergence should happen quickly but reach an optimal policy, as the wind farm owners would not desire a less-than-optimal policy. In the case of active wake control, if it takes a very long time to reach convergence, it is costly to run the program, and it might not be worthwhile for the wind farm to have the algorithm trained for it. Therefore, It is important to compare how fast convergence happens between QMIX and the other algorithms to determine whether it is a valid option.

3.3 Complexity

The number of wind turbines in the wind farm determines the complexity of the active wake control problem. Additionally, the problem can become more complex due to how the turbines are placed and the direction from which the wind is coming. Since FLORIS and TD3 take very long to run when the number of turbines grows to a large number of turbines, it is not feasible to get any results with limited time. For QMIX, it is then the question of whether it can get any results for large wind farms and if any improvements could be applied to improve the training process.

3.4 Improving Training

How well a deep reinforcement learning algorithm is trained can be measured in two ways: The value it converges at and how fast it converges at a value. An improvement in one of these factors, without negatively affecting the other, is an improvement in the training process. Here a few items are mentioned that could improve this process.

Since all states should be able to be explored while learning, an episode needs to run for a certain amount of time to be able to turn the turbines entirely. Since each turbine has a limit of turning 30 degrees in each direction from the wind, it takes 30 steps to reach the maximum with an angular velocity of 1.0. Therefore, a way to speed up training is to increase the angular velocity so that the turbines can reach their optimal yaw faster, which decreases the length of an episode. Increasing the angular velocity will considerably shorten training as simulating the environment is the most time-intensive process due to limited parallelization possibilities.

Lastly, training speed can be improved by using parallelization. The QMIX implementation can set the device to cuda, which will use the GPU of a computer by using PyTorch. More information on how PyTorch works can be found in [18]. Parallelization can significantly increase the speed of calculations needed for training neural networks, decreasing training time.

These are a few ways the algorithm's training is improved in terms of performance and speed. There may exist other ways the training could be improved, which could be explored in the future.

4 Experimental Setup

4.1 QMIX Implementation

A few QMIX implementations exist for different gym environments, such as StarCraft [19] and multi-agent particles [20]. The multi-agent particle environment was closer to the wind farm environment. Hence, the paper uses this implementation as adapting the algorithm to fit the current environment was less effort. The code provided by Steven Ho in [20] was relatively simple, making it easy to understand and adapt for this research. The repository has a few implementations consisting of a buffer for the batches, a network model, a training class, and some utility functions.

The buffer and utilities assist the training process and contain functions that the training uses during this process. The buffer stores the observations, actions, and rewards for each episode which can be sampled from the storage in batches to be used during training. In the utilities, there are a couple of functions, including Gaussian logarithm, LogSumExp and hard and soft update functions for updating the parameters of a neural network. The training file uses these functions, and nothing needs to be changed to make it work with the wind farm environment.

In the model, three classes exist using the neural network module from PyTorch to implement their neural network. The first is a policy network with a recurrent neural network, which aims to sample actions for its agent. Next, the model implements a recurrent neural network for approximating Q-values. This class can then be used to estimate the value of joint actions. Finally, the last class implements the overall architecture of the QMIX algorithm and estimates the joint action values. Putting it all together, these classes form the basis of the QMIX algorithm and allow for training and evaluation.

Moving on to the final part of the QMIX implementation, the training part. One class is located here, which is responsible for training the agents. The class contains multiple functions where the initialization sets up the networks for the actors, critics and the QMIX network. It is used to select actions for the agents depending on their observations, and the class inputs the observations into the neural networks. Furthermore, the reset function returns the hidden states to their initial values for the actors. Lastly, it updates the parameters for the neural networks and returns the critic and actor loss for monitoring purposes. The class trains the QMIX algorithm, but more is needed to make it work with an OpenAI gym environment.

An example of training is in a file where the QMIX algorithm is used for the multi-agent particle environment. Although a different environment is used in the paper, most of the code is still valid when adapted to the wind farm environment, especially the arguments used in the training loop. A few essential arguments are the number of episodes, length of the episodes, learning rates, batch size, gamma, tau, and a few more. Providing a way to change these is important and makes the training process more manageable. The structure of this training loop is almost entirely kept when it is changed to work with the wind farm environment.

4.2 Setup

In order to run experiments, the QMIX implementation was adapted to be used with the wind farm environment. The authors of [2] made the environment to research single-agent reinforcement learning. Therefore several functions had to be changed in the implementation of the environment to work with the chosen QMIX algorithm implementation. These changes included giving the observations as individual arrays per agent and splitting up the reward so the reward given is per agent. Initially, the reward is the total power output of all turbines, so to split it up, each agent gets the power output of the turbine they represent. The code in [21] was used to set up the experiments and get the results. With all the requirements, the experiments can be run that the following sections carry out. For the wind farm, some parameters can be adjusted, of which some values were changed and the rest left to their default values. A parameter that was adjusted was the turbine layout, as the different experiments have different turbine layouts. Another adjustment was the desired yaw boundaries to decrease the actions each turbine can take, possibly improving training. The action representation chosen was wind-based as this was the representation that performed the best in [2] and is when the direction of the wind determines the value of the yaw. Lastly, the maximum angular velocity is two degrees in some experiments to use a shorter episode length.

4.3 Hyperparameters

Since many hyperparameters can be adjusted, some exploratory experimentation was done by changing these values to get good results. Reinforcement learning is very sensitive to changes in the hyperparameters, so if one thing is changed a bit, the algorithm's performance could change drastically. The reason for choosing a specific value for a hyperparameter will be explained, and its impact on the results. A quick overview of the parameters and their values are given below that were chosen using exploratory experimentation:

- Maximum episodes: 10000
- Episode length: 25
- Policy learning rate: 0.00005
- Critic learning rate: 0.0005
- Alpha: 0.01
- Tau: 0.05
- Gamma: 0.99
- Batch size: 256
- Hidden dimensions: 128

The first two parameters that could be adjusted for training are the maximum number of episodes and the maximum length. As for the number of episodes, the required value is how many episodes until convergence, as there is no point in training anymore after this. Exploratory experimentation found that the algorithm converged on a policy after around four to five thousand episodes. However, this depends on the learning rate and the size of the wind farm. The environment is reset in each episode so the algorithm can see the same states again, which is crucial for its learning process. If the episode is short, the turbines do not have enough time to turn to the optimal angle as they move at a certain velocity. Since the maximum angle is 30 degrees from the wind direction on both sides, and the angular velocity is 2, it takes 15 steps to move to the maximum yaw. Since moving to the maximum yaw is rarely required, 25 steps were chosen as the episode length. Increasing this to more than 25 only increased the training time but did not have better performance.

Regarding the learning rate, QMIX has two learning rates that can be adjusted: the policy learning rate and the critic learning rate. The critic learning rate should be higher than the actor learning rate, as otherwise, the estimated Q-value will not correctly represent the value of the actions. Therefore, the critic learning rate was 0.0005, and the policy learning rate was 0.00005. These values were chosen as they converged at a relatively optimal result. However, the rate could be lowered more, and the number of episodes increased to get better results at the cost of training time.

Just as important are the alpha, tau and gamma values. Alpha is the policy entropy term coefficient and determines the exploration and exploitation trade-off. A high value promotes exploration, while a lower value promotes the exploitation of current knowledge. For the alpha value, 0.1 was chosen as this gave the best results because some exploration is done, but mainly the current information is exploited. The target network smoothing coefficient (Tau) decides the rate by which the target network parameters are updated—a smaller value results in updates being slower and, therefore, more stable learning. Next is the gamma value or the discount factor, which represents the choice of an agent between short-term and long-term rewards. A higher gamma value means the agents prioritize future rewards, and since, for the AWC problem, immediate rewards are less relevant, the value is placed at 0.99.

Finally, there are the batch size and hidden dimensions. There also exists a seed value to replicate the results. The hidden dimensions are set at 128, which should be enough to model the complexity for each agent as they only get three inputs. As for the batch size, it is placed at 256 because the program can run on a GPU to take advantage of parallel processing. This size could be increased but with a decreased learning rate as it converged too fast with the current values of the other parameters.

5 Results

Three versions of the main experiment were run, with the only change being the number of turbines and the layout. This change in the number of turbines and layout is supposed to answer the question of how well QMIX handles more complexity. For each layout, the performance and rate of convergence are observed to answer the other two sub-questions, which connects back to the main question. The following subsections highlight the results of the experiments.

5.1 3 Turbines

The algorithm was first tested on a 3-turbine layout as seen in Figure 2, where each black line represents a turbine, and the darker blue it is, the lower the wind speed. Each turbine is located 750m away from the others, around six rotor diameters. This simple layout of turbines in a row was also used in [2] with TD3 and FLORIS, and as such, the experiments can easily compare QMIX to these methods.

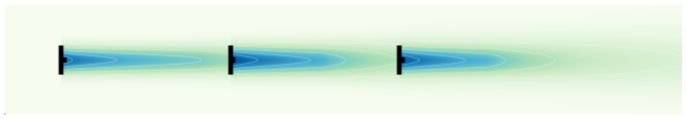

Figure 2: Layout of 3 turbines [22]

With the hyperparameters mentioned in the previous section, the program trained the QMIX algorithm for 10,000 episodes with the environment set up with three turbines. For simplicity, the experiments kept the wind direction from the west, and the wind speed was constant at 20m/s. The output of episodes was averaged over the last 50 episodes to smoothen out the graph to help gain a better insight into any trends in the chart. Figure 3 shows the average power output per episode against episodes. The baseline is when no control policy exists, and all turbines point toward the wind. FLORIS represents the value calculated by the FLORIS optimizer but is not necessarily the global optimum value. The QMIX algorithm was run ten times with different seeds, and the results averaged to ensure that a single run was not an outlier, shown in QMIX_avg. In addition to FLORIS, QMIX and the baseline, the training for TD3 was plotted in Figure 3. The graph shows that QMIX_avg converges at a value higher than the baseline but lower than the calculations by FLORIS. Compared to TD3, QMIX_avg converges at a later point than TD3, and TD3 converges at a higher point than QMIX. TD3 was run with the standard parameters provided by the implementation, so this could still be improved, but the same goes for QMIX. The general trend of the QMIX algorithm average power output per episode over episodes goes upwards, which shows it does learn over time, but it does not look like QMIX_avg has converged yet.

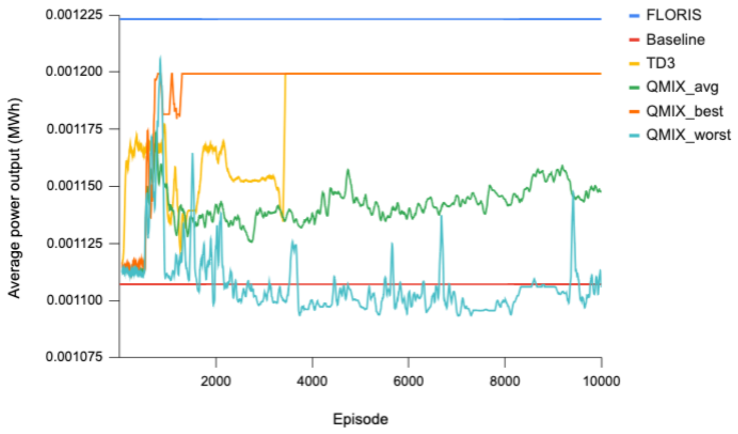

Figure 3: Average power output over 10000 episodes while learning for three turbines, using a moving average of 50

There was quite a lot of variation between the QMIX runs with different seeds. Three of the ten runs converged at the same value TD3 did, with the best run converging after around 1500 episodes as seen by QMIX_best in Figure 3. Another three runs learned a bit but did not converge at a value, while the last four runs stayed around the baseline where the worst result is plotted as QMIX_worst in Figure 3. A possible explanation of this discrepancy is a phenomenon known as catastrophic forgetting [23]. This issue could be improved in several ways, and section 6.3 discusses this occurrence further. Another factor might be that the hyperparameters are not optimal such as the learning rate not being at the correct value or the update rate of the target network not being set quite right. Since QMIX still learned an optimal result a few times, there is a high probability that the issue is a hyperparameter configuration.

5.2 16 Turbines

After observing some positive results for the three-turbine wind farm, the next step was to test the algorithm on a larger wind farm. A four-by-four grid consisting of 16 turbines was chosen where each turbine is spaced 750m away from the others, and the visual representation can be seen in Figure 4.

Similarly to the training on the 3-turbine wind farm, the QMIX algorithm was trained for 10,000 episodes with the environment set up with 16 turbines. Figure 5 shows the average power per episode against episodes for 16 turbines. Like the three-turbine experiment, the baseline is when no control

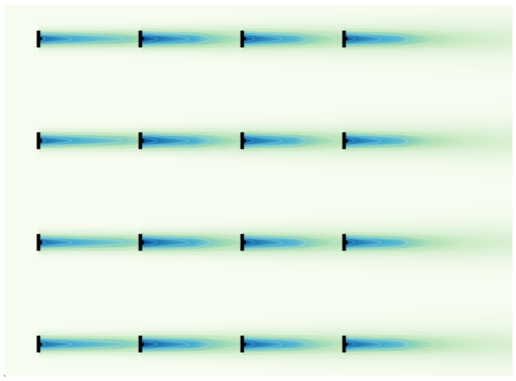

Figure 4: Layout of 16 turbines in a four-by-four grid [22]

policy exists, and all turbines point toward the wind. FLORIS can also still calculate a value and reaches an optimum average output of 0.0063124794MWh. Since QMIX takes much longer to run for 16 turbines, it was run three times with different seeds, and the results were averaged and plotted as seen in QMIX_avg. TD3 was also trained on the 16-turbine wind farm, and the results were plotted. The averaged results are better than the baseline but are significantly lower than the value TD3 converges at.

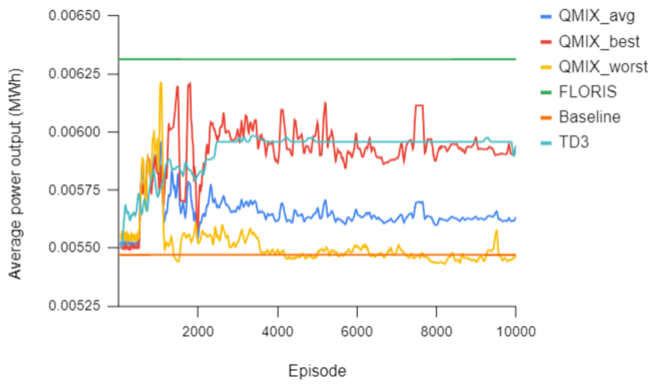

Figure 5: Average power output over 10000 episodes while learning for 16 turbines, using a moving average of 50

The same issue as with the three-turbine experiment occurs with the 16-turbine wind farm: a few runs do very well while the rest get mediocre results and stay near the baseline. Comparing the best run of QMIX with TD3, we see it converge at a similar point; however, QMIX has more fluctuations and does not converge to a single value. The worst run learns initially but then drops to around the baseline and stays there for the rest of the 10,000 episodes. This drop could happen for the same reasons mentioned in the three turbine experiment section, and since the problem is very similar, fixing it for one experiment has a high probability of fixing it for another.

5.3 Princess Amalia Wind Farm

The Princess Amalia Wind Farm is an existing wind farm located around 23km off the coast of IJmuiden consisting of 60 wind turbines. Figure 6 presents the layout using the gym environment with the exact coordinates retrieved from [24]. As the primary purpose of this research is to find out whether QMIX can be applied to active wake control, large existing wind farms should be experimented on to see if it could work in practice.

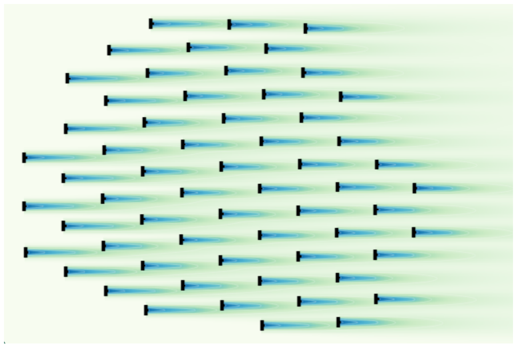

Figure 6: Layout of the Princess Amalia Wind Farm [22]

Training for the Princess Amalia Wind Farm takes a long time; therefore, the training time was reduced to 2400 episodes to see the initial learning curve, and the batch size decreased to 32 for QMIX. In Figure 7, the initial learning curve of the QMIX algorithm applied to the Princess Amalia Wind Farm is plotted. In addition to QMIX, the baseline was plotted, which is done using the same policy for the 3 and 16 wind turbines. Next, TD3 was also run for 2400 episodes and plotted to compare it with QMIX. Looking at the QMIX line, right after it starts learning, it drops to under the baseline and converges at a value of around 0.263MWh. TD3 performs much better than QMIX, and even though it drops under the baseline, around episode 2000, the average power output is above the baseline.

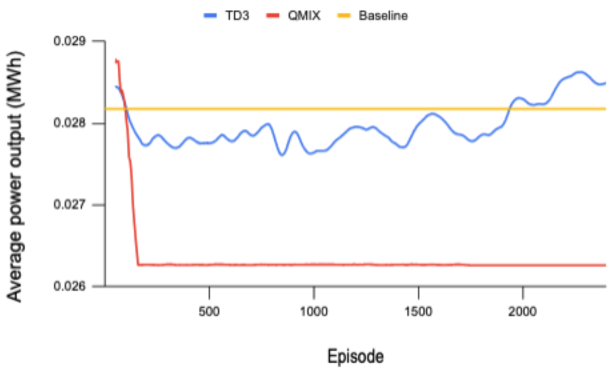

Figure 7: Average power output over 10000 episodes while learning for Princess Amalia Wind Farm, using a moving average of 50

Due to the long training times for the Princess Amalia Wind Farm, running the algorithms multiple times was not feasible to see if one run might be an outlier. The result of the QMIX training might therefore be an outlier, and for other runs, it might learn better. Having a large variety in results was also the case for the 3 and 16 turbine training runs, where some did very well and others did not. In order to fully say whether QMIX works for the Princess Amalia Wind Farm, it should be run multiple times, and section 6.3 recommends this for future research.

6 Discussion

This section looks at the research's limitations, followed by a discussion of the results with the limitations considered. It ends with recommendations on how the research can be extended or improved.

6.1 Limitations

As with any research, there are limitations to what can be done and the results that can be achieved. The most important limitations that affected this research are mentioned and explained below.

Time span

An important thing to consider is that the research took place over nine weeks, with the first week mainly consisting of deciding the research question. This duration limited the possibilities for experiments as even the experiments that were done now take a significant time to run. Therefore, the experiments could not deliver the optimal results or display the true capabilities of the QMIX algorithm. Training the QMIX algorithm can be parallelized; however, running the OpenAI gym environment can not, which means the time spent training is not something that can easily be improved. Therefore, the results of this research were highly dependent on the time spent on this paper.

Accuracy

Another limitation of the paper's results is the wind simulator's accuracy. In order to run wake calculations and do the simulation in a time so that it can run often, the model is simplified. Using this model means the results might not be as expected if the algorithm were applied to an existing wind farm with natural atmospheric properties. However, the models are accurate enough that when the optimal policies provided by the algorithms are applied to an existing wind farm, the results should be similar.

Implementation

Lastly, a factor that could have affected the experiments' results is the QMIX algorithm's implementation. It was assumed that the implementation works and is a valid implementation of QMIX. In the interest of time, the full implementation was not examined, so this could still be looked at in the future. If the implementation is wrong but gives positive results, it could become a variation of QMIX, or the results could be improved if the issues are fixed.

6.2 Results

With all the experiments run with the different wind farm layouts and the results observed, the three subquestions of the main research question of this paper can be answered. The trends across the different wind farm layouts will be observed and analysed.

Performance was the first component of whether the algorithm could be efficiently applied to the active wake control problem. FLORIS performed the best by far for the three and 16-turbine wind farms, so there is no reason to use TD3 or QMIX for these layouts in terms of performance. The average runs of QMIX performed worse than TD3, but this was because QMIX either converged to a good value or the baseline. The best runs of QMIX performed the same as TD3 and converged around the same value except for the Princess Amalia Wind Farm training. Since there is only one training run for this layout, no definite conclusions can be made on the performance of QMIX. To summarise, QMIX performs worse than FLORIS for the 3 and 16 turbine layouts but the same as TD3, which does better than the baseline. FLORIS cannot compute a value for the Princess Amalia Wind Farm, so it performs the worst, and QMIX performed worse than the baseline and TD3.

Convergence is the second aspect of the question and represents how much training time is needed to calculate a good policy for each algorithm. If the training time takes very long, it is very costly for the wind farm owners. FLORIS outperforms QMIX and TD3 in this area for the less complex wind farms as it calculates the policy in the first iteration but can not calculate a value within a reasonable timeframe for vast wind farms. Each episode takes around the same time with the same hardware for QMIX and TD3 as this is because giving the environment actions is the deciding factor for the length of an episode. For the three-turbine wind farm, QMIX does better than TD3 in terms of convergence and finds a policy after around 1500 episodes compared to TD3's 3500 episodes. If QMIX heads to the baseline, it does fluctuate a bit and does not fully converge. QMIX convergences at a similar rate as TD3 for the 16-turbine wind farm and faster for Princess Amalia but has a worse performance. Overall the rate of convergence and, therefore, the training time is better for QMIX than TD3 and worse compared to FLORIS for small wind farms.

Lastly, how the algorithm handles complexity was used to answer the main research question. Different wind farm layouts used the same experimental setup to test complexity. FLORIS can handle wind farms with 16 turbines but 60 turbines is too much, and it will not be able to calculate a value. QMIX and TD3 can calculate policies for all the wind farm layouts used, but they are not necessarily good policies. For the three and 16-turbine wind farms, QMIX and TD3 calculate a policy that gives around 8-9% more power output. Since the increase in power stays the same even though the layout complexity increases, it shows that the performance of QMIX and TD3 stays the same with an increase in wind turbines. Limited experimentation was done with the Princess Amalia Wind Farm, so it is possible that QMIX could calculate a policy that gives 8-9% more power output for other experiment configurations.

6.3 Future Work

Due to the limited time span of the research and how broad the subject is, more research can be done, and the current research can be extended.

More episodes
A simple way to extend this research is to run each experiment longer. Currently, each experiment is run for 10000 steps which are enough to allow it to converge, but if some parameters are changed, it could run a lot longer than that and might find a better policy.

Realistic wind conditions
Another point that could be extended is to use more realistic wind conditions. For the experiments conducted in this paper, the algorithms were trained on a constant wind direction and speed. A simple way to make the wind conditions more complex is to have either the wind move between a range of directions or have random changes in the wind speed. If the algorithm can be trained when there is an increase in the complexity of the wind conditions, the wind conditions at the wind farm it is trained on can be studied and then simulated in the environment. The overall goal of this research is to get a higher power output for wind farms. Since the wind conditions vary a lot in practice, the algorithms should be trained for many variations, although this might make the training time extremely long and too costly.

Larger wind farms
The QMIX algorithm was tested on the Princess Amalia Wind Farm for a limited period. This experiment was to observe whether anything would be learned for such a complex wind farm. More experiments could be run on the Princess Amalia Wind Farm or any other existing wind farm to confirm that QMIX can handle complex wind farms.

Catastrophic forgetting
A problem that deep reinforcement learning faces while continuously learning is: that knowledge of the previously learned tasks is forgotten when learning information for the current task. This process is known as catastrophic forgetting [23] and was observed in many experiments in this paper. Since the QMIX implementation uses batches, a way to solve this is first to fill the sample memory with random results so that it does not forget and get a terrible result, even though it might still forget a bit. There may exist other ways to solve this problem, such as by changing the hyperparameters or augmenting the algorithm, and further research can look at this.

Hyperparameter tuning
Lastly, better hyperparameter tuning could be done to extend this research. The full range of values that could be used was not fully explored, and other combinations of values may provide excellent results. Exploration could be done in multiple ways, such as randomized search and grid search, although these methods take very long and will take even longer with the algorithm's runtime. Research could then be done to develop a better way of exploring the hyperparameters for this algorithm being used for the active wake control problem.

7 Responsible Research

This section reviews the experimental setup (Section 4) and looks at the reproducibility of the results. Additionally, the scientific integrity of the research is considered, and the ethics of the research are discussed.

7.1 Reproducibility

For this paper, reproducibility was a goal to ensure that other scientists could get the same results that were achieved in this paper. The ability to reproduce the results was done by uploading the code onto GitHub and ensuring instructions were left behind so anyone could follow the same steps used in the paper to run the program. Additionally, all the parameters used are provided, and a seed was used such that if all the parameters are the same, the results should be the same. The running time, however, will depend on the hardware used.

7.2 Scientific integrity

This research involved collaboration with J. Yeh, G. van der Schaaf, I. Plamadeala, and M. Filimon. They tested other multi-agent reinforcement learning algorithms for the active wake control problem, and some discussions were held on their findings. All research from external sources has been cited.

7.3 Ethics

The outcome of this research is positive for society because if the results are promising, wind farms can produce more energy. More energy from wind increases the amount of sustainable energy available, which is positive for society. No personal data is used, and the experiments were all simulated on a computer, so there are no ethical issues.

8 Conclusion

This paper aimed to determine whether the QMIX algorithm can be efficiently applied to the problem of active wake control. The research was done by comparing the performance, rate of convergence and ability to handle complexity with other algorithms or methods that have been applied previously. Additionally, if multi-agent reinforcement learning techniques are better than single-agent reinforcement learning, this problem should also be answered. Looking at the results and the analysis, QMIX has many variations in its results, but the best results outperform TD3. With better hyperparameter tuning and possibly other improvements, QMIX could outperform TD3 consistently in terms of performance and in training time.

QMIX showed promising results in this research; therefore, multi-agent reinforcement learning algorithms are a promising method to outperform their single-agent counterparts for the problem of active wake control. Many other multi-agent reinforcement learning algorithms exist that might outperform QMIX, and research could be done to compare different multi-agent algorithms for the problem. In the same sense, different single-agent reinforcement learning algorithms may perform better than TD3, which could also be researched.

Solving the active wake control problem is challenging, and more research will have to be done to solve it. This paper highlights one possible method, and although it gave some positive results, more work will need to be done for it to be applied effectively in practice.

References

[1] N.-J. Vermeer, J. Sørensen, and A. Crespo, "Wind turbine wake aerodynamics," *Progress in Aerospace Sciences - PROG AEROSP SCI*, vol. 39, pp. 467–510, 10 2003.

[2] G. Neustroev, S. P. E. Andringa, R. A. Verzijlbergh, and M. M. De Weerdt, "Deep reinforcement learning for active wake control," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '22. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2022, p. 944–953.

[3] R. Barthelmie, S. Pryor, S. Frandsen, K. S. Hansen, J. Schepers, K. Rados, W. Schlez, A. Neubert, L. Jensen, and S. Neckelmann, "Quantifying the impact of wind turbine wakes on power output at offshore wind farms," *Journal of Atmospheric and Oceanic Technology - J ATMOS OCEAN TECHNOL*, vol. 27, pp. 1302–1317, 08 2010.

[4] M. Jacobson and M. Delucchi, "A path to sustainable energy by 2030," *Scientific American*, vol. 301, pp. 58–65, 11 2009.

[5] S. Boersma, B. Doekemeijer, P. Gebraad, P. Fleming, J. Annoni, A. Scholbrock, J. Frederik, and J.-W. van Wingerden, "A tutorial on control-oriented modeling and control of wind farms," in *2017 American Control Conference (ACC)*, 2017, pp. 1–18.

[6] 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, 06 2021.

[7] P. Stanfel, K. Johnson, C. 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, p. 043305, 07 2021.

[8] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning, ieee transactions on systems," *Man and Cybernetics. Part C: Applications and Reviews*, vol. 38, pp. 164–181, 01 2008.

[9] M. Littman, "Value-function reinforcement learning in markov games," *Cognitive Systems Research*, vol. 2, pp. 55–66, 04 2001.

[10] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," 07 2000.

[11] T. Rashid, M. Samvelyan, C. Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," 03 2018.

[12] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2020.

[13] Z. Deng, C. Xu, X. Han, Z. Cheng, and F. Xue, "Decentralized yaw optimization for maximizing wind farm production based on deep reinforcement learning," *Energy Conversion and Management*, vol. 286, p. 117031, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890423003771

[14] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[15] NREL, "Nrel/floris: A controls-oriented engineering wake model." [Online]. Available: https://github.com/NREL/floris

[16] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1587–1596. [Online]. Available: https://proceedings.mlr.press/v80/fujimoto18a.html

[17] 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 *ICLR*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15

[18] [Online]. Available: https://pytorch.org/

[19] Oxwhirl, "GitHub - oxwhirl/pymarl: Python Multi-Agent Reinforcement Learning framework." [Online]. Available: https://github.com/oxwhirl/pymarl

[20] Steven-Ho, "madrl-baselines/entrance.py at master · Steven-Ho/madrl-baselines." [Online]. Available: https://github.com/Steven-Ho/madrl-baselines

[21] [Online]. Available: https://github.com/schobbejak/QMIX-Active-Wake-Control

[22] [Online]. Available: https://github.com/Algtudelft/wind-farm-env

[23] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1611835114

[24] "Offshorewind RVO," https://offshorewind.rvo.nl/files/view/acdaee43-5600-4899-8515-dcf113de70e5/1576834165hkn_20191217_project.