Multi Agent Deep Deterministic Policy Gradient for Active Wake Control

Guus van der Schaaf

Supervisor(s): Mathijs de Weerdt, Greg Neustroev

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: Guus van der Schaaf
Final project course: CSE3000 Research Project
Thesis committee: Mathijs de Weerdt, Greg Neustroev, Przemyslaw Pawelczak

Abstract

In wind farms wind turbines are often placed close to each other. Each turbine generates a turbulent wake field, this field negatively affects subsequent turbines. This can cost more than 12% efficiency. To decrease this loss we can steer the turbines away from the wind direction, this will decrease the individual turbine power output, but can increase the total power output of the farm. As the size of the farm increases the number of possible actions increase exponentially. Due to this a numerical solution is not feasible. A reinforcement learning technique has been proven useful in the past, but a standard single agent implementation is still very computationally expensive. We evaluate the effectiveness of MADDPG on the active wake control problem. MADDPG is a multi agent reinforcement learning algorithm. MADDPG will be compared to the numerical solver FLORIS and to the already implemented and proven TD3 (which is a variation on a single agent DDPG algorithm). We compare the eventual output power of the algorithms with MADDPG. From the results we can see that MADDPG does improve on the learning performance of TD3, but since MADDPG needs to manage more neural networks the overhead is larger. MADDPG reaches an optimum solution in less training steps, but these steps take significantly more time.

1 Introduction

In wind farms turbines are placed close to each other, this gives the most performance per square meter.

This close placement does however introduce wake effects. The wake effects are the result of turbulent air generated by the wind turbines rotor blades. The air moves slower behind the rotor, this negatively influences the next turbine in the row [1].

If we could address these losses we can improve the power output and thus generate more renewable energy. A study showed that in Denmark the observed loss in performance was 12% due to wake effects[2]. With bigger wind farms these losses become even larger.

These losses could be addressed by steering the turbines. A wind turbine generates maximum power when steered straight in the direction of the wind. By steering a turbine away from the wind we reduce the power of this turbine, but can greatly improve the output of the next turbine [3, 4]. This is called active wake steering [5]. The improved overall efficiency can be achieved without building any new infrastructure so it is an almost free improvement.

The wakes can be simulated using simplified mathematical models of the wakes [6, 7, 8]. FLORIS (Flow Redirection and Induction in Steady-State)[9] is a state of the art numerical solver. Due to the continuous action and state space this problem quickly becomes too big for a numerical approach.

In the past a solution was explored with Deep Reinforcement Learning [10]. This booked some great result, but this also suffered from the exponential explosion of parameters.

To adress these limitations we experiment with a Multi Agent Reinforcement Learning (MARL) approach. We hope to improve on the single agent learning approach by shrinking the action space to a simple action per agent. Our hypothesis is that a multi agent reinforcement learning approach will perform better on large windfarms and suffer less from the combinatorial explosion. We assume that Multi-agent DDPG is good algorithm to solve this problem. MADDPG is a well established algorithm where decentralized agents learn a centralized critic, this leads to policies that only use local information of the agents.

In section 2 we explain the main differences between traditional RL and MARL and we introduce the MADDPG algorithm. In section 3 we explain the setup for the experiments and present the results. In section 5 we highlight the responsibility and ethics of the research. In section 6 we discuss the results and highlight some research areas that might be interesting for new research.

2 Multi Agent Reinforcement Learning

Traditional Reinforcement Learning has been a established machine learning method for years [11]. The main learning loop of RL uses the following general structure:

1. The agent observes the environment
2. With this observation the agent computes an action to execute
3. After the agent executes this action the agent gets a reward from the environment
4. With this reward we can train the agent to maximize its reward

By shaping the reward we can guide the agent to a goal. In the basic approach we save all observation and future reward information in a so called Q-Table. This works great for smaller state-action spaces.

Nowadays a Deep Q-Learning [12] approach is more popular, where the Q-Table is approximated with a Neural Network, this saves creating a fully populated table and thus saves exploring the complete state space.

In the Actor-Critic [13] paradigm the agents 'brain' is split in two parts, the Actor and the Critic part. The Actor is a function that has as input the current observation and outputs the best action to take. The Critic is a function that has as input a observation and an action and outputs the 'value' of this action in this given state. The value of the action can be seen as the future possible reward gained from this action.

Some problems are better suited for a multi agent approach, instead of a single overarching agent controlling everything there are now multiple agents controlling only their own actions, each agent has its own 'brain'. A main advantage of the multi agent architecture is that this shrinks the action space of each agent. Instead of computing x^n actions we now have to compute $x * n$ actions. We could also shape the reward function differently such that the agents will 'fight' each other in a competitive scenario or work together in a cooperative scenario. This also leads to a more elegant approach to some problems. For the active wake problem a cooperative algorithm is the best fit.

The multi agent paradigm also introduces some problems. For instance since all agents execute their own actions with their own policies there is no stationary policy. We might get a situation where the agents are chasing around the goal instead of towards the goal. Also the runtime complexity of MARL can be higher. By having each agent act on its own there is more overhead.

Some examples of Deep Cooperative Multi Agent algorithms are [14]:

- Independent Q-Learning
- Distributed Q-Learning
- Hysteric Q-Learning
- Multi Agent Deep Deterministic Policy Gradient
- COMA

In MARL the algorithms can be divided in some categories. One of those categories is Independent Learners. Independent Q, Distributed Q and Hysteric Q are all part of this category. With Independent Learners each agent is modeled as its own completely independent agent only using its own observations for its policy. Another category is Fully Observable Critic. In this architecture the critic(s) can see all policies of all agents. This stabilizes the learning since there is a global stationary policy. COMA and MADDPG are part of this category.

We use the Multi Agent Deep Deterministic Policy Gradient algorithm. This algorithm is based around the Actor Critic architecture. There are N centralized critics with N decentralized actors, as illustrated in figure 1. COMA is very similar to MADDPG, but COMA uses one central critic for all agents. We assume that by having N critics each agent can learn a more specific policy. The execution loop of MADDPG

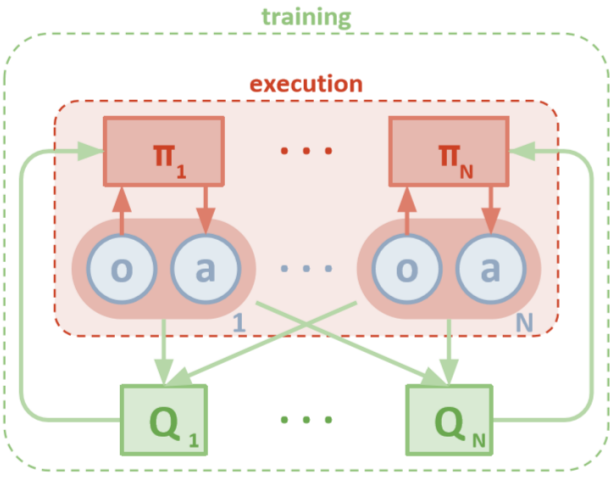

Figure 1: Architecture of MADDPG [15]

uses the following structure:

1. Each agent observes its local environment.
2. Each agent uses its own local observation to generate an action to execute.
3. Each agent gets a reward and a new observation from the environment. Each agent saves these values in its replay memory.
4. We execute the training function for each agent

The training function is described as follows for each agent:

1. Take a minibatch from the replay memory. (environment states, next environment states, taken action, observed reward)
2. Get from all agents the action they would have executed in these states.
3. Get from all agents the action they would have executed for the next states.
4. Calculate the critic value for all actions that would have been taken in these states.
5. Similar to normal q learning we discount these values with the rewards. $critic\_target = reward + \gamma * critic\_value$
6. The critic network can now be updated by taking a gradient step in this direction
7. For the actor network we calculate the current policy action for the agent in the given states
8. We calculate the critic value for the actions of all agents and the new action of this agent.
9. With this critic value we can update the actor network.

From the pseudo-code it can be seen that during execution time we do not need to communicate with the other agents. MADDPG uses a extra actor and critic network called the target actor and target critic. The target networks are used when a value is used for the update process, the normal network is updated each training step. After some training steps we do a polyak update [16] to the target network.

3 Experimental Setup

Traditional Reinforcement Learning tasks are often ran using the OpenAI Gym framework [17], this gives a standardised base to create environments and algorithms. The wind farm experiment also has been converted to a OpenAI Gym [18]. This is used as a base for our experiments.

For evaluating our research question we need to test with multiple sizes of wind farms. This way we can verify on smaller models that our algorithm converges and test the scaling of the algorithm on the larger farms. We used the following wind farm layouts.

- 3 turbines in a row, this is called a wind tunnel. This is to verify our solution is correct.
- A grid of 4x4 turbines, this is a lot larger to test the scaling.
- A model of the Princess Amalia Wind farm [19] consisting of 64 turbines. This is a model of an actual wind farm, this test will tell us if the algorithm could be used in the real world.

We tested multiple algorithms versus MADDPG to verify that it actually resulted in correct solutions and to compare as a baseline to other approaches.

- FLORIS [20], the numerical solution (this was only ran on the 3 and 8 turbine farm)
- TD3, as implemented in this paper [10] on the Active Wake Control problem
- MADDPG

All experiments logged data in a tensorboard format [21]. With this setup we could easily in real time monitor the performance and track what the algorithms were doing. With this data we could observe if a algorithm was converging to a solution. All experiments were tested with a constant wind speed of 20km/h from the west.

4 Results

A result can be considered good if it reaches a optimum goal in a less training steps. However we also need to evaluate the amount of time each algorithm needs per training step.

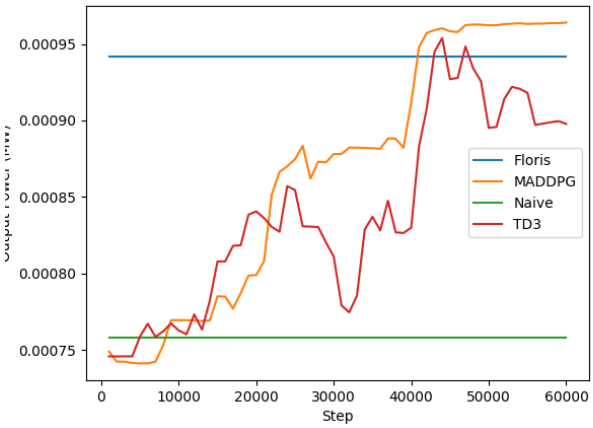

Figure 2: Output Power on the 3 turbine layout evolving over learning time

In the 3 turbine experiment depicted in figure 2 we can observe that MADDPG works. It learns a strategy that is better than doing nothing. Eventually it meets the optimal target set by FLORIS. We can also observe that for small windfarms a multi agent approach does not give big benefits since it does not reach the optimal value in less training steps.

For 16 turbines we can see in figure 3 that MADDPG learns faster than TD3. In this run it did learn a inferior strategy and dropped. By running the algorithm multiple times this could be alleviated. This can be explored in the future. With more learning time both algorithms would probably reach the optimum.

On the Amalia wind farm we can see that MADDPG learns faster. It reaches a value of 0.14 about 50000 training steps earlier. And stays for almost the complete training time above the value of TD3.

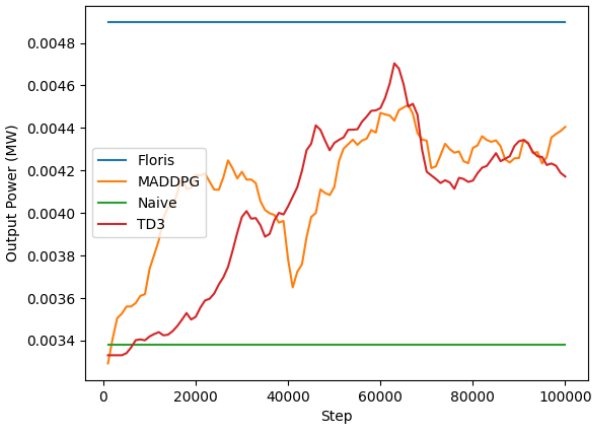

Figure 3: Output Power on the 16 tubine layout evolving over learning time

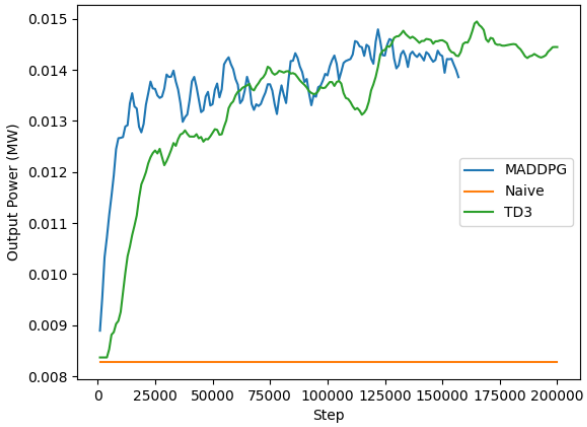

Figure 4: Output Power on the Amalia layout evolving over learning time

We also logged the time of every full time step the time data can be seen in table 1. The corrected rows have their times divided by the number of turbines in the farm. From this we can observe that all tested algorithms scale linearly in time per step with the number of turbines. And MADDPG needs 7x more time per time step. These values are dependent on the used hardware.

Algorithm	3 turbine	4x4 grid	Amalia
TD3	11ms	31ms	217ms
MADDPG	82ms	418ms	1680ms
TD3 Corrected	4ms	2ms	4ms
MADDPG Corrected	27ms	26ms	28ms

Table 1: Training time per algorithm on the Amalia wind farm

5 Responsible Research

All code used to create the MADDPG algorithm is uploaded on GitHub [22]. To ensure reproducibility of our research we use seeds to initialize the random processes. This ensures that the random process will output the same order of values every time, by using the same seed one should get an identical learning curve for the same hyperparameters.

The OpenAI Gym used in our research is based upon FLORIS Simulator, this simulator has been proven to be an almost perfect simulator for this experiment. FLORIS has been compared with a particle simulator and produced approximately the same results in much quicker time. This ensures that the data is accurate enough.

6 Conclusions and Future Work

Multi Agent Deep Deterministic Policy Gradient Reinforcement Learning is an effective method to get a solution to the active wake control problem. The problem is very hard to solve, mainly since the state action space is exponential.

We first evaluated if MADDPG in the most simple implementation could generate useful results. This was proven to be very effective. MADDPG beats the baselines we set for it:

- Doing nothing, the naive approach. This is a very useful baseline to beat, since we now know that at least we have not made things worse.
- FLORIS, if we meet the quality of FLORIS we have at least an almost optimal solution. FLORIS cannot run on the bigger wind farms so this baseline only holds for the basic farms.
- TD3, a implementation of single agent DDPG.

We can see that we can meet all these baseline for the three experiment setups.

However the expected scaling of the algorithm did not happen. Since there are a lot more calculations per time step the training process takes significantly longer. We do reach the optimum in a lower number of steps, but the steps take an order of magnitude longer to process. This might however be solved by using a faster computer.

In the future MADDPG can be improved in multiple ways. One could for instance change the state space per agent. A useful property of MADDPG is that the agents do not necessarily need the same state information, thus we could give each agent only information that is deemed relevant. For instance only giving the agents the yaws of turbines in their vicinity. This could shrink the state space and thus increase the learning speed.

One could also change the reward function. In our algorithm the reward is the total power output of the farm. This could be split in a reward per agent, this would involve calculating the contribution of that single agent in the whole farm, the result would be that the algorithm could be steered to the correct solution more quickly.

There are also multiple extensions of MADDPG [14] page 11. MADDPG-GCPN has been proven to improve on MADDPG. MAAC has proven to be useful to reduce the dimensionality of the observations.

References

[1] LJ Vermeer, Jens Norkaer Sorensen, and Antonio Crespo. "Wind turbine wake aerodynamics". In: *Progress in aerospace sciences* 39.6-7 (2003), pp. 467–510.

[2] R Barthelmie et al. "Modelling the impact of wakes on power output at Nysted and Horns Rev". In: *European wind energy conference*. Vol. 2. 2009, pp. 1351–1373.

[3] Pieter MO Gebraad et al. "Wind plant power optimization through yaw control using a parametric model for wake effects—a CFD simulation study". In: *Wind Energy* 19.1 (2016), pp. 95–114.

[4] Michael F Howland, Sanjiva K Lele, and John O Dabiri. "Wind farm power optimization through wake steering". In: *Proceedings of the National Academy of Sciences* 116.29 (2019), pp. 14495–14500.

[5] Jan Willem Wagenaar, L Machielse, and J Schepers. "Controlling wind in ECN's scaled wind farm". In: *Proc. Europe Premier Wind Energy Event* 1.01 (2012).

[6] Antonio Crespo, J Herna, et al. "Turbulence characteristics in wind-turbine wakes". In: *Journal of wind engineering and industrial aerodynamics* 61.1 (1996), pp. 71–85.

[7] Niels Otto Jensen. *A note on wind generator interaction*. Vol. 2411. Citeseer, 1983.

[8] Angel Jimenez, Antonio Crespo, and Emilio Migoya. "Application of a LES technique to characterize the wake deflection of a wind turbine in yaw". In: *Wind energy* 13.6 (2010), pp. 559–572.

[9] Jennifer Annoni et al. "Analysis of control-oriented wake modeling tools using lidar field results". In: *Wind Energy Science* 3.2 (2018), pp. 819–831.

[10] G. Neustroev et al. "Deep Reinforcement Learning for Active Wake Control". In: 2022, p. 10.

[11] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[12] Jianqing Fan et al. "A Theoretical Analysis of Deep Q-Learning". In: *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. Ed. by Alexandre M. Bayen et al. Vol. 120. Proceedings of Machine Learning Research. PMLR, Oct. 2020, pp. 486–489. URL: https://proceedings.mlr.press/v120/yang20a.html.

[13] Ivo Grondman et al. "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 1291–1307. DOI: 10.1109/TSMCC.2012.2218595.

[14] A. Oroojlooy and D. Hajinezhad. "A review of cooperative multi-agent deep reinforcement learning". In: *Applied Intelligence*. 2022.

[15] Ryan Lowe et al. "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *Advances in neural information processing systems* 30 (2017).

[16] Taisuke Kobayashi and Wendyam Eric Lionel Ilboudo. "T-soft update of target network for deep reinforcement learning". In: *Neural Networks* 136 (2021), pp. 63–71.

[17] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[18] G. Neustroev et al. *The Wind Farm Gym*. 2022. URL: https://github.com/AlgTUDelft/wind-farm-env.

[19] Noord Zee Loket. *Prinses Amalia Windpark*. 2008. URL: https://www.noordzeeloket.nl/en/functions-and-use/offshore-wind-energy/free-passage-shared-use/hollandse-kust-noord-wind-farm-zone-including/.

[20] NREL. *FLORIS Version 2.4*. 2021. URL: https://github.com/NREL/floris.

[21] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[22] G. van der Schaaf. *MADDPG Code*. 2023. URL: https://github.com/JeffersonYeh/MARL-awc-windfarm/tree/maddpg.