Project Report

Machine Learning and Optimisation in Optimising Wind Farm Layout Design Under Uncertainty

Abstract

With an increasing reliance on renewable energy, there is an increasing demand for wind farms. Often, wind farms are built such that wind turbines are placed in sub-optimal locations respective to other nearby wind turbines. This project uses machine learning techniques in order to optimise Wind Farm layouts for minimising cost and maximising power output in order to determine the optimal locations of wind turbines within a wind farm. The result is multiple optimal wind farm layouts. By using a dataset of already existing wind data, machine learning is used to make a probability distribution to model the wind. The Jensen Wake model is used to approximate the wake effect, which is used to calculate the expected power output. And finally testing the multi-objective optimization algorithms; NSGA2, NSGA3 and AGE-MOEA, against each other in order to create a set of optimal solutions for different price points.

I certify that all material in this dissertation which is not my own work has been identified.

Signed: Tom Boatman

Contents

1 Introduction 2

2 Literature Review and Project Specification 2
2.1 Literature Review Summary 2
2.1.1 In search of flexible and robust wind farm layouts considering wind state uncertainty 2
2.1.2 Wind farm layout optimization under uncertainty 3
2.1.3 Multivariate and Multimodal Wind Distribution Model Based on Kernel Density Estimation 3
2.2 Project Specification 3

3 Design 4
3.1 Gaussian Mixture Model 4
3.2 Wake Model 5
3.2.1 On the application of the Jensen wake model using a turbulence-dependent wake decay coefficient: the Sexbierum case 5
3.3 Multi-Objective Optimisation 6

4 Development 6
4.1 Wind Data Collection and Interpretation 6
4.2 Gaussian Mixture Model 8
4.3 Wake Model 10
4.4 Representing a Wind Farm Layout 10
4.5 Multi-Objective Optimisation 11

5 Testing 12
5.1 Gaussian Mixture Model 12
5.2 Wake Model 13
5.3 Multi-Objective Optimisation 14

6 Description of the final product 16
6.1 Using the final product 17

7 Analysis of the final product 18
7.1 Requirements 18
7.2 Results 18
7.3 Performance and Accuracy 19

8 Critical assessment of the project as a whole 20

9 Conclusion 21

1 Introduction

Due to the over reliance of non-renewable energy sources, which, as of 2019, supplied up to 80% of the world's energy [1] and polluted the atmosphere and furthering the greenhouse effect, there has been a global shift towards relying more on clean and renewable energy sources. Wind power is the second most common renewable energy source in the world, producing about 24% of the world's renewable energy [2], and the demand for wind power is rapidly growing. Estimates from 2019 expect onshore wind output to rise by 57% and offshore is expected to triple by 2024 [3].

This project is aimed at optimising wind farms considering the cost and power output of a wind farm. Although adding more wind turbines in a wind farm will generate more electricity, It will also increase the monetary cost of the wind farm. However this is not a linear relationship; the wake effect is the main hurdle and it negatively affects the power output of a wind farm. It is an interaction between wind and wind turbines where a wind turbine will create a wind shade downwind, negatively affecting how much power other turbines that are inside the wind shade can produce [4]. By positioning the wind turbines in different locations in a given area, the wake effect could potentially be reduced, and thus causing an increase in power output.

To calculate the wake effect, you would need to know certain details of the wind turbines such as the height, blade length and power coefficient, as well as the speed and direction of the wind [5]. However there is no way to predict future wind conditions with complete accuracy. By using a mixture model, it is possible to model past wind conditions and create a probability distribution for the wind speed and wind direction in order to estimate expected wind conditions.

For a given amount of wind turbines, there would be a different optimal solution for each differing number of wind turbines, and as a result, optimisation techniques must be used to calculate a set of optimal solutions for different price points. A multi-objective optimisation algorithm would have to calculate multiple scenarios of wind conditions, turbine placements and use the probability distribution and wake effect to create a set of optimal solutions.

The aim of this project is to use the techniques mentioned above to optimise wind farms layouts in order to maximise potential increases in the power output while minimising the cost of a wind farm.

2 Literature Review and Project Specification

2.1 Literature Review Summary

This is aimed at reviewing papers that are similar to my project and seeing what I can learn from them or looking at projects which demonstrate things that are useful and essential to my project.

2.1.1 In search of flexible and robust wind farm layouts considering wind state uncertainty [6]

This paper is very similar to this project, using a wind dataset to optimise a wind farm layout for 2 objectives; power output and cost, using a pareto front to find the most optimal solutions. It

uses the frequency of particular wind speeds and wind directions to analyse the wind data. Then, they use approximately 100 different scenarios of wind direction and wind speed to capture the variation in wind conditions, ignoring other values such as air density. Vertical wind movement is not considered either, and the hub height is constant. The cost is proportional to the number of turbines as all the turbines are the same.

For the wake model, they used the Jenson wake model to calculate the wake effect and compute the reduction in wind velocity. It uses NSGA-II (or NSGA2) for genetic multi-objective machine learning to produce a best case and worst case set of pareto fronts.

2.1.2 Wind farm layout optimization under uncertainty [7]

This paper focuses on optimising wind farms under uncertainty while still taking the monetary cost into consideration, however, unlike my project, it does not rely on a pre-existing dataset to produce data, nor does it use a genetic algorithm to optimise the solution. Instead it uses iterative matrices in order to optimise wind farms.

Due to the lack of pre-existing wind data, the solutions are rather generic, and as a result it cannot be tailored to a specific geographical area, making it more of a demonstration on how to find optimal wind layouts rather than finding specific optimal layouts for a real location.

This paper also focuses on different hub heights to optimise the layout further, taking the cost of the different hub heights into consideration. Although this increases how optimal the set of solutions are, it does require more computational power and more complexity in its algorithm.

2.1.3 Multivariate and Multimodal Wind Distribution Model Based on Kernel Density Estimation [8]

This paper does not directly optimise the layout of wind farms, instead, it focuses on analysing wind data so that it can be used for optimising a wind farm. The wind data it used has been obtained from the North Dakota Agricultural Weather Network (NDAWN) [9] and uses the wind speed, wind direction and air temperature, from 2000-2009. Wind speed & direction are measured at 3 metres above the soil surface, and the recorded value is the average of all hourly average wind speeds and directions for a 24-hour period from midnight to midnight.

It uses a Multivariate and Multimodal Wind distribution model. It applies this to 3 cases: a Univariate model to analyse only the wind speed, in which they represent it using different probability distributions. The second case is a Bivariate model for the wind speed and wind direction, which is represented using a contour plot and probability density. The final case is a Multivariate model for wind speed, wind direction and air density, where the wind speed and air density are used to calculate the wind power density. In case 2, the air density is estimated using the temperature, the height above sea level (512m) and some additional constants.

2.2 Project Specification

The project is split into 3 main parts; The Gaussian Mixture Model, the Wake Model and the Multi-Objective Optimization algorithm.

The aim of the Gaussian Mixture Model is to optimise the wind farm for a certain location using a dataset of wind data from an existing wind farm at that location or a nearby location. The

Gaussian Mixture Model will map the existing dataset in order to create a mathematical model that assesses the likelihood of certain wind conditions. The parameters of the wind data are wind speed and wind direction. These are mapped against each other in a 2D array and graph for the machine learning algorithm to create the Gaussian Mixture Model.

The wake model will calculate the power output of the wind farm. It will need 3 input parameters; the wind speed, the wind direction and the wind farm layout. It uses these 3 parameters to model the wake effect, and then uses tha t and a power equation to calculate the expected power output from all of the turbines in the wind farm.

The Multi-Objective Optimization algorithm will aim to create a pareto front of solutions for wind farm layouts. A 2-dimensional Pareto front is a set of non-dominated solutions, where each solution is not beaten by any other solution for both of the objectives. Each solution in the pareto front is considered equally good. The Multi-Objective Optimization algorithm is often a genetic algorithm that uses a set of solutions and tests them against the objectives to find the best ones, often creating new solutions by randomly changing existing solutions and combining existing solutions, amongst other methods. In this case, each solution would be a wind farm layout while the 2 objectives are expected power output and cost. The cost would be dependent on the number of turbines. To calculate the expected power output, the Gaussian Mixture Model would be used to weight multiple wind conditions, each of which would use the wake model to calculate the expected power output.

3 Design

3.1 Gaussian Mixture Model & Wind Data Requirements

A Gaussian Mixture model is a probabilistic model consisting of multiple Gaussian distributions. Given a 2D dataset and a number of finite Gaussian distributions, and under the assumption that all data points belong to one of a given number of Gaussian distributions, it uses machine learning to calculate the Gaussian distributions and assign each datapoint to one of those distributions.

The Gaussian Mixture model will be dependent on scikit-learn for the main algorithms. Scikit-learn [10] is an open-source, commercially usable, external python module for machine learning. It will need a dataset of existing wind data to create the Mixture model.

A Gaussian Mixture Model uses unsupervised learning, which means it does not need users to help or supervise the generation of the model. It is one of the most accurate and widely used unsupervised mixture modelling methods, and is the fastest algorithm on scikit-learn for learning mixture models, and has no biases towards zero or certain cluster sizes.

Paper 2.1.3 [8] includes wind temperature, which when combined with altitude above sea level, allows them to estimate wind density. Including this in my Gaussian Mixture Model would likely give small benefits to accuracy and will require significantly more computational power and time to produce an accurate mixture model making it infeasible for the scale of this project. As a result, I will just be using wind speed and wind direction to produce a 2D Gaussian Mixture Model.

The log law is used in papers 2.1.2 [7] and 2.1.3 [8] to calculate the different wind speeds at different wind heights. For my project, I intend on using the same model and height wind turbine for each wind turbine, and because my data was collected from a wind turbine, It will be at the relevant height, so I don't need the log law to calculate the expected wind speed.

3.2 Wake Model

Wind Turbines generate electricity from oncoming wind. Each turbine creates a wake downwind that reduces the energy generated from any subsequent downwind turbines. This is called the wake effect. A Wake model is a mathematical model designed to simulate or calculate the wake effect. There are multiple different models designed to model the wake effect for wind farms. The Jensen model, the Larsen model and Fuga [5] are good for large wind farms because they are robust and don't require a significant amount of computational power.
The Jensen model is one of the most popular wake models and is also the model used in Paper 2.1.1 [6]. It was first described by N.O. Jensen in 1983 in his paper “A note on wind generator interaction” [11].

3.2.1 On the application of the Jensen wake model using a turbulence-dependent wake decay coefficient: the Sexbierum case [12]

This paper uses an onshore wind farm to test the Jensen wake model against other wake models and also observing the wake decay coefficient. It contains a diagram showing the wake model, which can be seen in Figure 1, and the following equation describing the wind speed deficit (1),

$$1 - \frac{u_2}{u_1} = \frac{u_1 - u_1\sqrt{1 - C_t}}{(1 + k_w\, x/r_r\,)_2} \tag{1}$$

where u_1 and u_2 are the initial wind speed and downwind wind speed respectively, C_t is the trust coefficient, r_r is the rotor radius, k_w is the wake decay coefficient and x is the downwind turbine distance. The paper explains that for combining multiple wakes at a turbine, the square of the resulting wind speed deficit is the sum of the squares of each individual wind speed deficit.

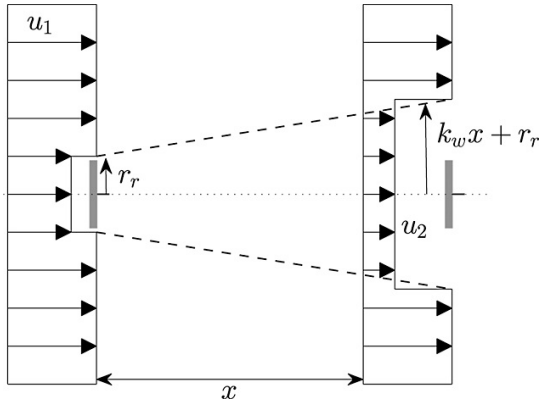


Figure 1: Wake effect from a turbine

The paper “On-line monitoring of power curves” [13], uses the following equation (2) in order to compute the power output from a single turbine.

$$P = 0.5\rho\pi R^2 C_P\,(\lambda, \beta) V^3 \tag{2}$$

ρ is the air density, R is the wind turbine blade length $C_P(\lambda, \beta)$ is the power coefficient, and V is the wind velocity at the turbine.

3.3 Multi-Objective Optimisation

This project requires optimising for multiple objectives: The power output and the cost of the turbines. The cost of the turbines will be proportional to the number of turbines, this assumption was also made in paper 2.1.1 [6].

Rather than trying to calculate and find a single optimal solution, it would be more beneficial to find a set of optimal solutions, then use subjective decision making to choose a solution, that way, more options are being considered. The set of solutions would be determined by a Pareto Front. A Pareto Front is a subset of points within a larger set, where each datapoint in the set is considered equally good. Assuming we have 2 objectives, then for each data point in a dataset, if it is beaten by another data point in both objectives, then it is considered dominated and is not included in the Pareto Front. Every non-dominated data point is included in the Pareto Front. Each item in the Pareto Front cannot be distinctly better than another without any loss of sorts, thus requiring a subjective decision, often decided by considering the budget.

Many Multi-Objective Optimisation algorithms exist. The popular ones include NSGA2, NSGA3, MOEA/D, AGE-MOEA and C-TAEA. The algorithms will create a set of optimal solutions in the form of a Pareto Front. Paper 2.1.1 [6] uses NSGA2 for its multi-objective optimisation. NSGA2 [14] and NSGA3 are two of the most popular genetic multi-objective sorting algorithms, because of their fast and elite multi-objective genetic algorithm. NSGA3 is designed to handle many more objectives than NSGA2, but that doesnt matter for this as I only have 2 Objectives. AGE-MOEA is similar to NSGA2 but it tries to estimate the shape of the Pareto Front.

4 Development

The project is coded using python 3.9.1 [15] due to the vast library availability and support.

4.1 Wind Data Collection and Interpretation

The dataset I am using is from the La Haute Borne Wind farm in France, published by Energie Renewables [16]. It contains 2 datasets; one from 2013-2016, and another from 2017-2020, however the data in the 2017-2020 dataset is incomplete. The data is collected from 4 wind turbines, which are the same model, so they can be expected to produce the same power output given the same wind conditions. Each datapoint in the datasets has approximately 110

different values for different attributes. The required attributes are Ws_avg and Wa_avg, which are the average wind speed and average wind angle/direction respectively. Having written a python program to filter through the data, making sure the data is logical (for example, checking the wind angle is between 0 and 360), and plotting the wind speed against the wind direction/angle. I did this for both the 2013-2016 dataset, which can be seen in figure 2, and the 2017-2020 dataset, which can be seen in figure 3. The dataset in figure 3 is much smaller because it is incomplete. Although it is named 2017-2020, the data ends in mid-January 2018.

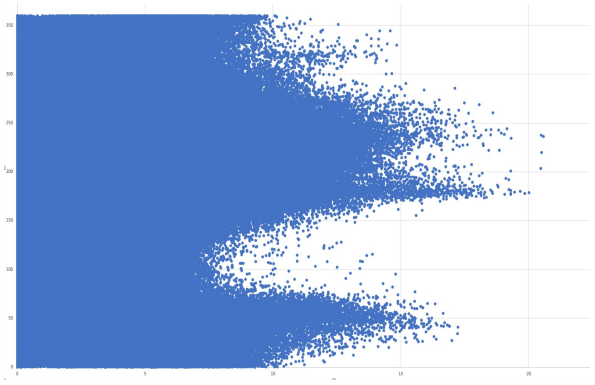


Figure 2: 2013-2016 wind dataset

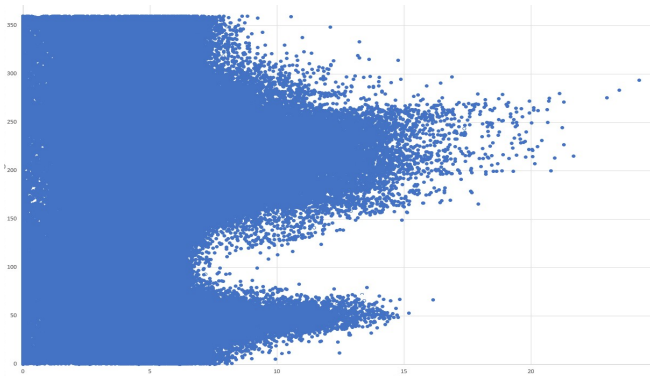


Figure 3: 2017-2020 wind dataset

After making different graphs for each year 2013-2018, I now can see how It changes for the years that there is data for. Figure 4 contains wind data from 2013, Figure 5 has 2014's wind data, Figure 6 is 2015, Figure 7 is 2016, Figure 8 is 2017 and Figure 9 is 2018.

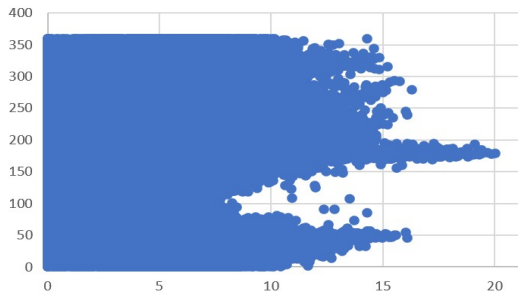


Figure 4: 2013 wind data

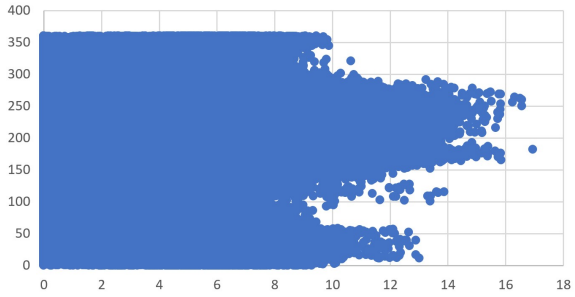


Figure 5: 2014 wind data

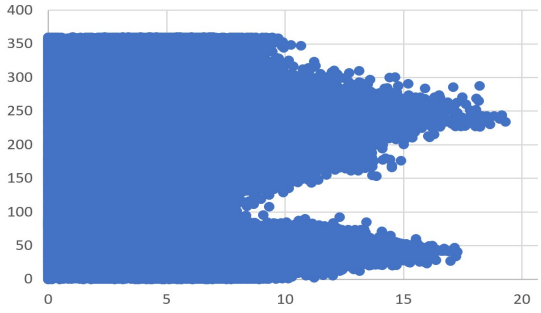


Figure 6: 2015 wind data

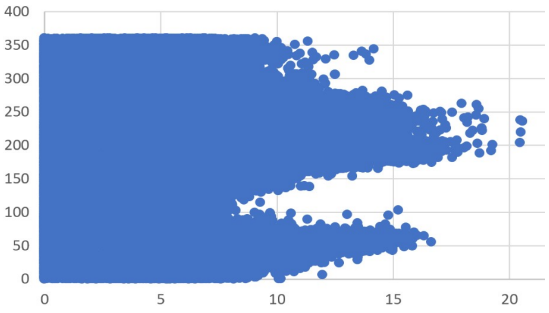


Figure 7: 2016 wind data

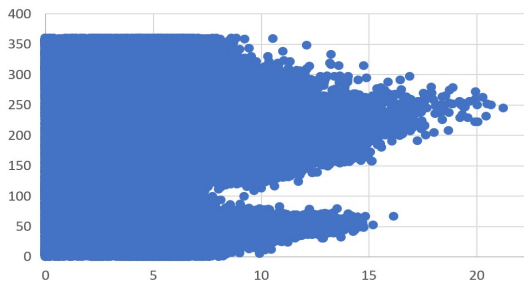


Figure 8: 2017 wind data

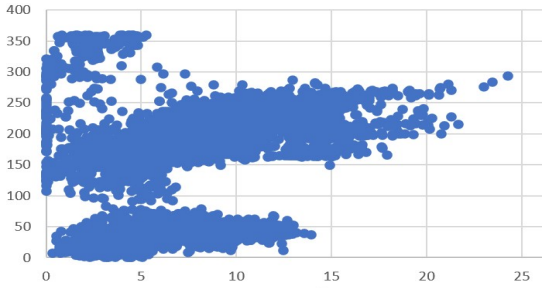


Figure 9: 2018 wind data

All of these graphs have peaks around 50° and 200°-250° and troughs around 100° and 300°. This means that the 2 peaks are approximately opposite directions, as are the 2 troughs. Because there is significantly less data in 2018, Figure 8 shows how the data tends to show how the wind speed and direction correlates and how it clusters into groups that can be calculated using a mixture model. 2013 seems to have different peaks in the wind data compared to the remaining years. While not confirmed, I speculate that the weather was different during that year. For creating the Gaussian Mixture Model, I will use data from 2013, 2014, 2015, 2016 and 2017.

4.2 Gaussian Mixture Model

The first thing that is needed is to find the number of Gaussian distributions that is needed for the project. I could use trial and error to find the best fit. It is also possible and feasible to create a graph showing the density of the dataset so that I can accurately tell how many distributions it has. Figure 10 shows the density of the dataset., where white has 0 density and black is the highest density.

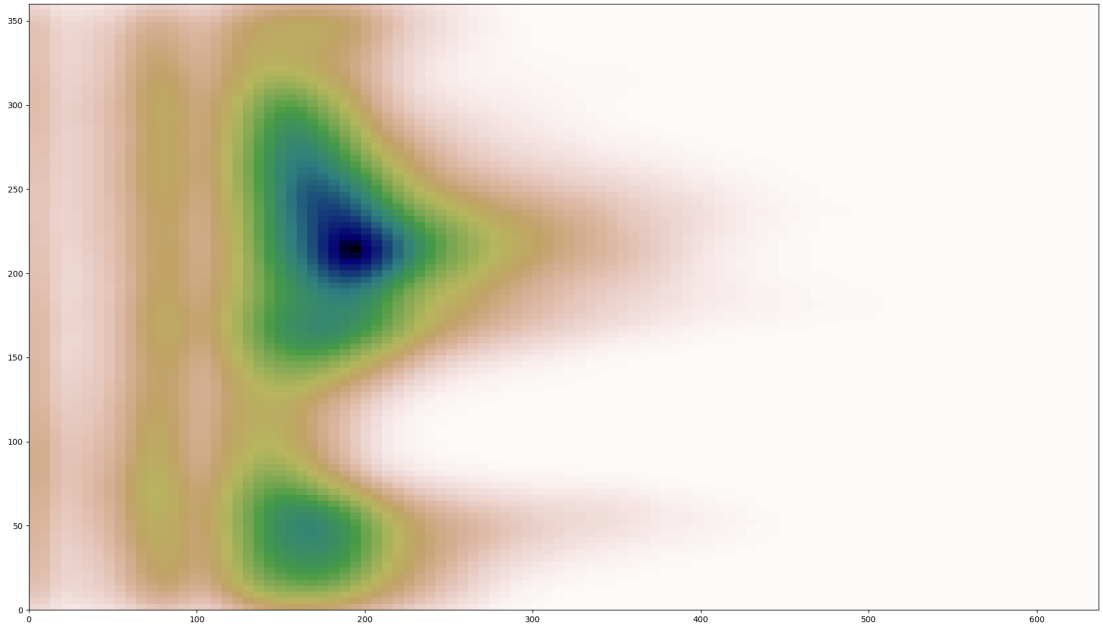


Figure 10: Heatmap of the wind dataset

For representing the wind data for a Gaussian Mixture Model to use, there is an issue with simply plotting the wind speed against wind direction. Because the wind direction is continuous and loops around every 360°, the Gaussian Mixture Model cannot interpret this, which adds some unneeded inaccuracy to the Model. To deal with this, instead of representing it using cartesian coordinates, it will need to be represented using polar coordinates. Rather than using 2 regular perpendicular axes, a polar coordinate system uses an angle and a magnitude to plot each data point. This is ideal as the wind direction is an angle and the wind speed is a positive magnitude. After plotting a dataset using polar coordinates, the plane can be labelled and represented using a cartesian coordinate system in which the y-axis contains the north-south wind component and the x-axis contains the east-west wind component. For comparison, Figure 11 shows the density of the wind dataset in polar coordinates.

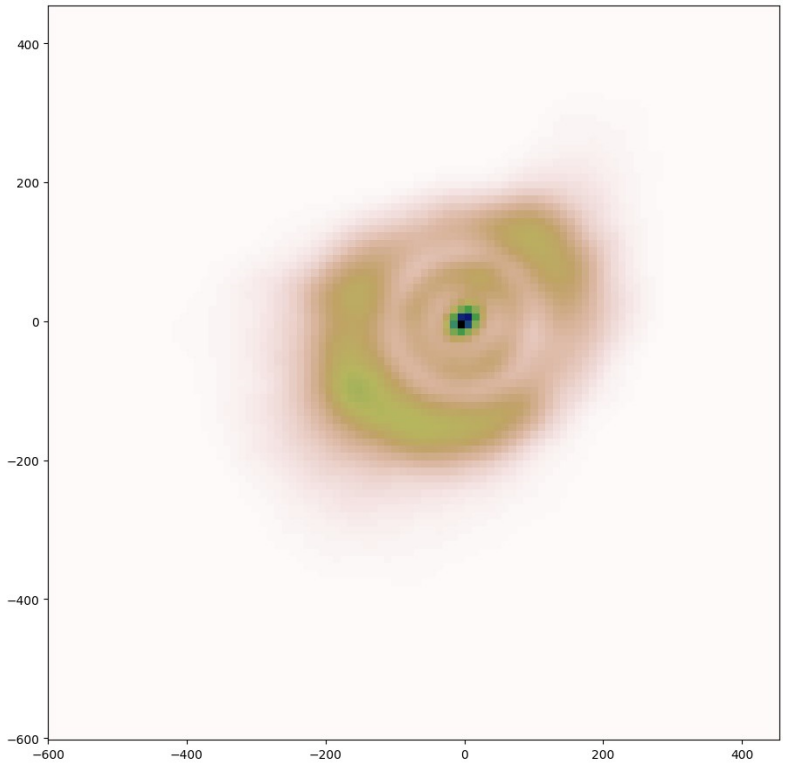


Figure 11: Heatmap of the wind dataset using polar coordinates

I can use the Scikit Learn [10] external python library to implement a gaussian mixture model. This allows me to easily use the gaussian mixture model without spending a tremendous amount of time implementing it. It also prevents me from making mistakes when implementing the mixture model myself.

In order to find the number of Gaussian distributions for the wind dataset, I used scikit learn to create multiple Gaussian Mixture Models for a range of Gaussian distributions per model. Figure 12 shows Gaussian Mixture Models, for 2, 3, 4 and 5 Gaussian distributions respectively.

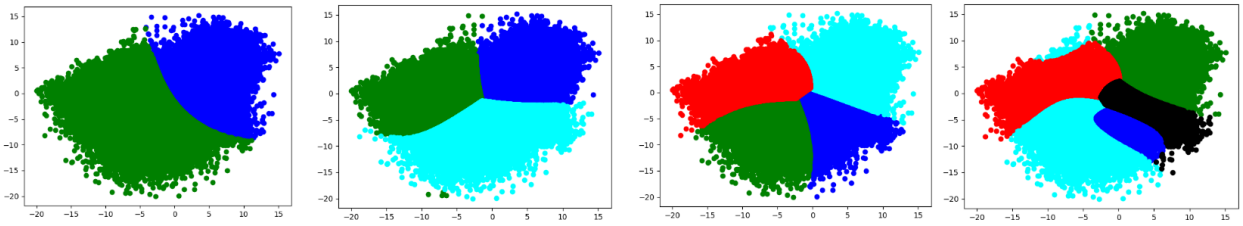


Figure 12: GMMs representing the wind data for 2, 3, 4 and 5 Gaussian distributions.

I can use scikit learn's silhouette score to assess the accuracy of the accuracy of the Gaussian Mixture Model. Silhouette score gives a score of 0 to 1 with 0 being the worst possible score and 1 being a perfect score. The Gaussian Mixture Model with 2 Gaussian distributions was found to have a silhouette score of 0.40 (rounded to 2 decimal places). The Gaussian Mixture Model with 3 Gaussian distributions had a silhouette score of 0.39, the one with 4 Gaussian distributions had a silhouette score of 0.36, and 5 Gaussian distributions had a score of 0.33. It was clear that the accuracy was declining as the number of Gaussian distributions increased, so I decided I should use 2 Gaussian distributions since it had the best silhouette score, meaning that it is the most accurate number of Gaussian distributions to represent the wind dataset.

4.3 Wake Model

To calculate the wake model and power output, some constants that are described in section 3.2 are needed. These are the Thrust Coefficient, wake decay coefficient, the air density, power coefficient and the wind turbine blade length. The wind turbine blade length for the turbines that gathered the data in the wind dataset [16] are 41m, so I shall use that. Paper 3.2.1 [12], states that the Thrust Coefficient is 0.75, at wind speeds within the range 6.8–10 ms^{-1}, so I shall use this as a large portion of my wind data lies within this range. The paper also shows that out of the wake decay coefficients that they tested for, for a single wake, the value of 0.038 showed the smallest root mean square error, so I will use that value. The air density is 1.225 kg/m^3 for dry air at sea level [17]. The power coefficient peaks at 0.44 [18], so I shall use that value.

I coded the wake model in python without any external library. The only imported module from the standard library was math [19] because it contains the maths needed for the angles required for calculating the wake model. It was implemented in a separate python file from the main part of the programme and uses subroutines which are imported into the main program and are accessed and used there.

With the mathematics required for calculating the wake model for each turbine and the power output having been coded in python, the program needs to determine if a wind speed deficit from the wake needs to be calculated. To achieve this, for each turbine, the program must check if each other turbine lies within its wake. Figure 1 shows that for downwind distance x from a turbine, a second turbine should be equal or less than $k_w x + r_r$ perpendicular to the downwind direction from the initial turbine.

4.4 Representing a Wind Farm Layout

The wind turbines that were used to gather the data have a hub height of 80m and a rotor diameter of 82m [16]. The recommended minimal distance between turbines is considered to be 3 times the rotor diameter [20], making the minimal distance between turbines 246m. Both papers 2.1.1 [6] and paper 2.1.2 [7] have a minimal distance less than 246m.

For a multi-objective optimization algorithm to optimise wind farm layouts for both cost and power output, the wind farm layout needs to be represented. The wind farm itself is a regular square grid of potential positions where a wind turbine can be placed, where the distance between 2 adjacent potential turbine positions is 246m. The paper discussed in 2.1.2 [7] used a 10x10 grid to describe the wind farm, and because anything higher was too computationally expensive, I shall also describe each wind farm layout using a 10x10 grid. This means that the wind farm being represented by the grid will have a total area of 2.46km^2. One of the simplest ways to represent a 10x10 grid is using a list of 100 boolean values, where each 0 represents an absence of a turbine in that position, and each 1 means that there is a turbine in its respective position.

4.5 Multi-Objective Optimisation

The multi-objective optimization is coded using an external python library called pymoo [21]. This contains many multi-objective optimization algorithms including NSGA2, NSGA3 and AGE-MOEA. These are all genetic algorithms which means they follow the diagram shown in Figure 13. They all start off with an Initial Population, which uses a sampling method to generate a population, where each item in the population is a wind farm layout. The sampling method for this is called random sampling, which randomly sets each boolean in the 100 long list to either a 0 or 1. Evaluation involves executing the problem, which in this case is calculating the power output and the cost. The power output calculations are outlined in sections 3.2 and 4.3. The cost is dependent on the number of wind turbines, however it is not linear. The paper titled “Offshore Wind Farm Layout Optimization Using Adapted Genetic Algorithm: A Different Perspective” [22], uses the following equation (3) to calculate the cost of the wind farm, where N is the number of turbines.

$$Cost = N\left(\frac{2}{3} + \frac{1}{3}e^{-0.00174N^2}\right) \quad (3)$$

Survival uses the evaluation to determine which of the population will survive and be kept, and which of them will be removed. Then selection selects members of the population to be parents for the crossover, the selection can either be random or use a type of tournament selection. The crossover combines parents in order to create one or more offspring. In this case, I use binary two point crossover for the wind farm layouts, which will select some parts of one parent and some parts of the other parent to produce new wind farm layouts. Mutation randomly changes the new children to add some additional diversity to the population. The wind farm layouts use bitflip mutation, which has a small chance to flip each bit. The algorithm will end once a certain criteria has been met, in this case, as there is no real limit to the optimisation algorithm, I must use a generation limit which stops the algorithm after a given number of generations.

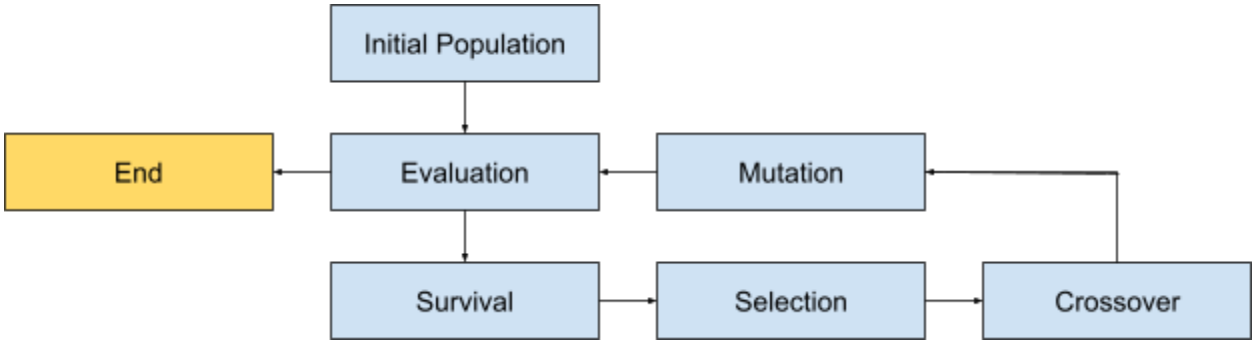


Figure 13: Basis of a genetic algorithm

The optimisation algorithms either try to minimise all objectives or maximise all objectives, however one of my objectives needs to be minimised (cost) and the other should be maximised (power output), I ended up minimising the function and providing the multi-objective optimization algorithm the negative of the power output in order to deal with this issue.

5 Testing

My code uses a readme file which contains many parameters and constants that can be changed. The program will read these parameters and constants from the readme file. This improved ease of use for the code, which helped testing.

5.1 Gaussian Mixture Model

Figure 14 shows the dataset split into 2 Gaussian Mixture Models, one represented by green, and the other by blue. To use the Gaussian Mixture Model, I take a uniform sample of each wind speeds and wind directions, then use the model to calculate the probability of each of the sampled wind speeds and wind directions. The probability will then be used to create a weighted average power output for a given wind farm, in order to create an expected power output. Figure 15 shows a sample of 3 different wind speeds with 9 different wind directions, and Table 1 shows the probability of the uniform sample. With no clear errors in the data produced and everything making logical sense, I can infer that the Gaussian Mixture Model is working as intended.

	0°	40°	80°	120°	160°	200°	240°	280°	320°
6 m/s	0.1234	0.1244	0.1218	0.0706	0.1209	0.1230	0.1192	0.0920	0.1046
12 m/s	0.1164	0.1164	0.1163	0.1078	0.1164	0.1164	0.1162	0.1122	0.0820
18 m/s	0.1132	0.1132	0.1132	0.1129	0.1132	0.1132	0.1131	0.1129	0.0952

Table 1

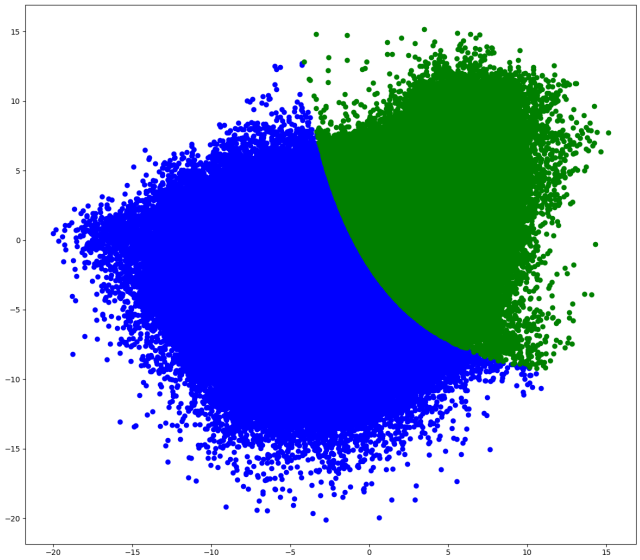


Figure 14: Gaussian Mixture Model

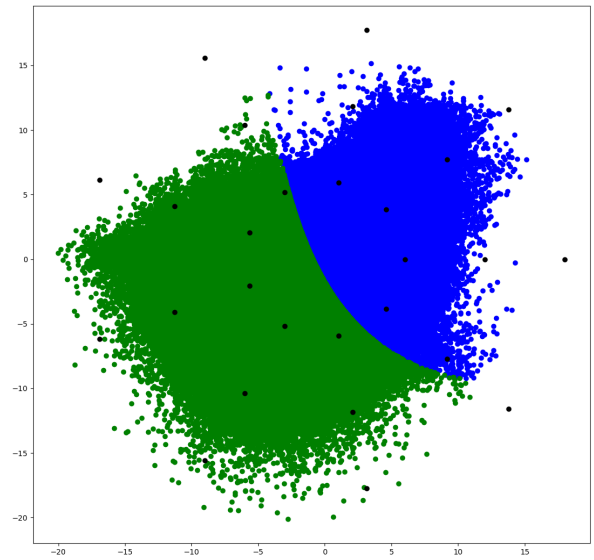


Figure 15: Gaussian Mixture Model with uniform sampled wind conditions

5.2 Wake Model

I ran some tests for the wake model function to return the power output. The results are shown in Table 2. All power outputs are calculated to 4 decimal places.

Test no.	Wind speed (m/s)	Wind direction (degrees)	Number of wind turbines	Description of wind farm layout	Power output
1	0	0,45,90,135,180, 360,450	0	No wind turbines	0
2	0	0,45,90,135,180, 360,450	1	The single turbine was tested in multiple positions	0
3	0	0,45,90,135,180, 360,450	100	All positions in the 10x10 grid has a wind turbine	0
4	0-20	0	0	No wind turbines	0
5	0-20	90	0	No wind turbines	0
6	1	0,45,90,135,180, 360,450	1	The single turbine was tested in multiple positions	1423.2342
7	2	0,45,90,135,180, 360,450	1	The single turbine was tested in multiple positions	11385.8732
8	5	0,45,90,135,180, 360,450	1	The single turbine was tested in multiple positions	177904.2686
9	10	0,45,90,135,180, 360,450	1	The single turbine was tested in multiple positions	1423234.1491

10	1	0,90,180,270	100	All positions in the 10x10 grid has a wind turbine	35602.6907
11	1	45,135,225,315	100	All positions in the 10x10 grid has a wind turbine	58701.9931
12	5	0	10	The turbines are in a vertical line (north-south)	445033.6332
13	5	90	10	The turbines are in a vertical line (north-south)	1779042.6863
14	5	0	10	The turbines are in a horizontal line (east-west)	1779042.6863
15	5	90	10	The turbines are in a horizontal line (east-west)	445033.6332
16	5	90, 180	19	An L shape combining a horizontal line and vertical line of turbine	2046172.0509

Table 2

Tests 1, 2 and 3 show that the power output should be 0 when the wind speed is 0, and tests 1, 4 and 5 show that the power output is 0 when the number of wind turbines is 0. Tests 6, 7, 8 and 9 show that the power output increases with the wind speed as the power output equation (2) describes. Tests 6, 10 and 11 show the effect of the wake model. Without the wake effect, tests 10 and 11 should produce 100 times more power than test 6, with the same wind speed and 100 times more wind turbines, however, test 10 only had 25 times more power output than test 6 (This demonstrates that the maximum percentage power loss caused by the wake effect is 75%), and test 11 had 41 times more power output than test 6. This is the expected outcome because the wake effect will reduce the wind output for most of the turbines, and shows that the wake model is working. Additionally, despite having the same wind speed and the same uniform wind farm layout, test 11 has a higher power output than test 10, this is expected because with a square grid, when the wind is coming in from a 45 degree angle, not only do a lot of turbines have less turbines in their wake, but the downwind distance between subsequent turbines is larger, thus reducing the wind speed deficit from the wake effect and increasing the power output. Tests 12 and 15 show that the wake effect is working for a row of 10 turbines parallel to the wind direction and are in contrast to tests 13 and 14 which have a row of 10 turbines perpendicular to the wind direction, where there should be no turbine lying within the wake of another turbine. This is also evident by test 13 and 14 having a power output exactly 10 times larger than test 8, which has the same wind speed and 1 turbine. In conclusion all these tests show the wake effect and power output is working as intended.

5.3 Multi-Objective Optimisation

Hypervolume is used to assess the pareto front created by the optimisation algorithm. With 2 objectives, It is calculated by finding the dominated area of the objective space. In order to test different multi-objective optimisation algorithms, the hypervolume should be plotted against the number of generations for each algorithm. I tested 3 different popular Multi-Objective

Optimisation algorithms against each other for my project, they were each tested with a population size of 400 and terminated after 800 generations. Figure 16 shows the pareto fronts of NSGA2 (left), NSGA3 (middle) and AGE-MOEA (right). All the pareto fronts appear to have approximately the same shape, however the NSGA3 pareto front has significantly less solutions. When looking at the solutions given by each algorithm, I found that NSGA2 gave 172 solutions, NSGA3 gave 43 solutions and AGE-MOEA gave 90 solutions. NSGA3 producing fewer solutions is not ideal and provides less options for a potential end user. However, NSGA2 and AGE-MOEA produced multiple unique solutions each for some given number of turbines, this isn't necessarily an issue but it isn't ideal either as it would require additional decision making for an end user. An example of this can be found in solutions A and B in Figure 19 in section 7.2. This occurred more often in the results generated using NSGA2 than AGE-MOEA, which is why NSGA2 has significantly more solutions.

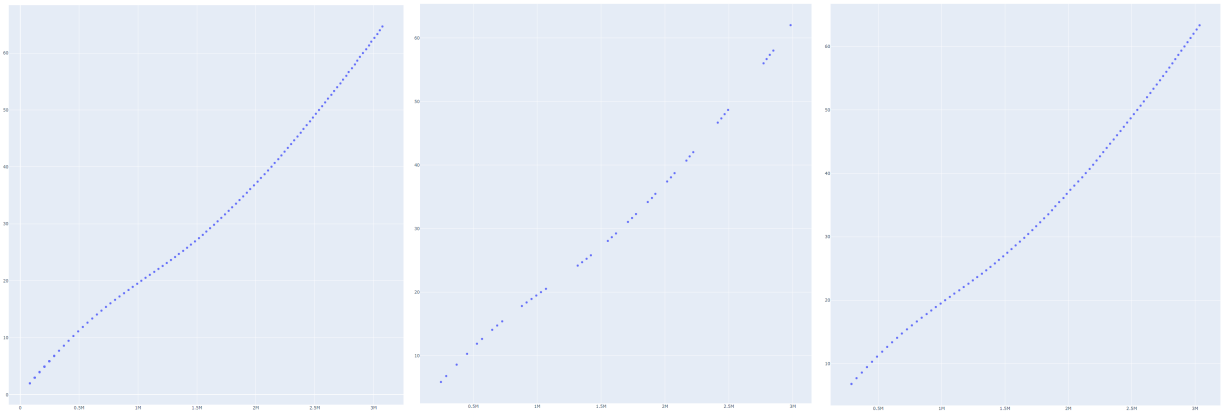


Figure 16: Pareto Fronts created by NSGA2, NSGA3 and AGE/MOEA

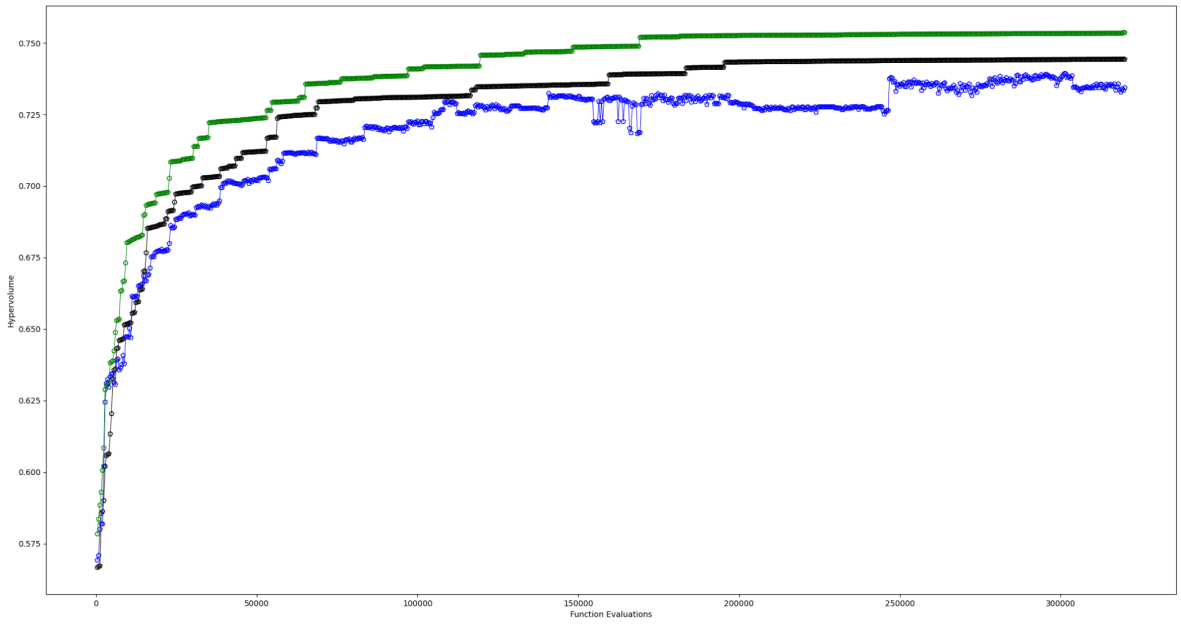


Figure 17: Hypervolumes of NSGA2, NSGA3 and AGE/MOEA

Figure 17 shows the hypervolume of each algorithm, where NSGA2 is black, NSGA3 is blue and AGE-MOEA is green. Overall, it is shown from this that for this project, AGE-MOEA is the most efficient Multi-Objective Optimisation algorithm as for the vast majority of its run time, AGE-MOEA had a larger hypervolume, and also had the largest hypervolume at time of termination. NSGA2 is shown to be the least efficient for the same reasons. The solutions found for AGE-MOEA are shown and assessed in 7.2 Results.

6 Description of the final product

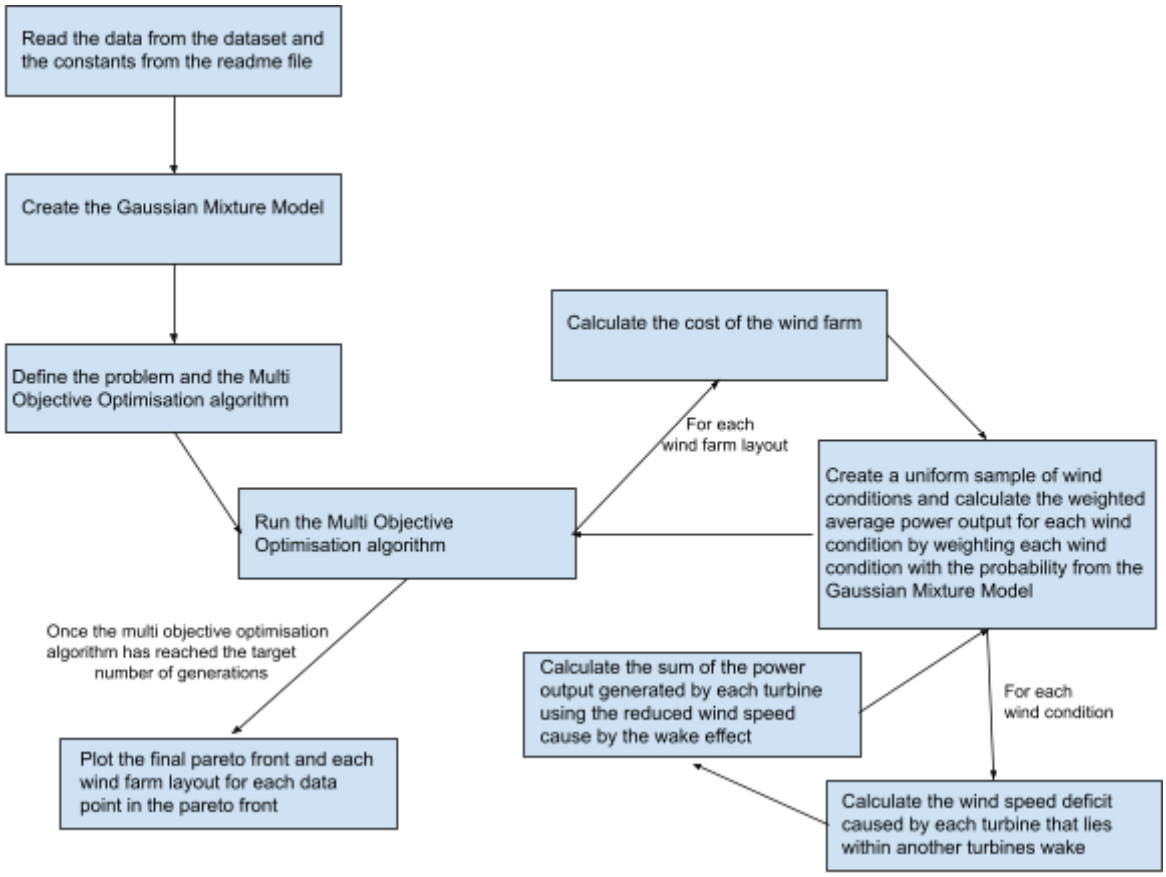


Figure 18: Overview of the programme

The project is made up of 3 main parts; the Gaussian Mixture Model, the wake model and the Multi-Objective Optimisation algorithm. What they do and how they work are mostly described in sections 3, 4 and 5. Figure 18 shows the process that the programme takes when it runs.

Out of the 3 main parts, the Gaussian Mixture model was implemented using SciKit Learn [10] and the Multi-Objective Optimisation Algorithm was implemented using Pymoo [21]. I coded the wake model and power output myself. Below is the pseudocode for calculating the power output of a given wind farm, wind speed and wind direction.

FUNCTION WAKE MODEL:

PASS IN: LIST: Wind Farm Layout, FLOAT: wind speed, FLOAT: wind direction

```
        Thrust Coefficient = 0.75
        Wake Decay Coefficient = 0.038
        Power Coefficient = 0.44
        Air Density = 1.225
        Minimum Distance Between Turbines = 246
        Turbine Radius = 41
        TYPE LIST: Turbines
        FOR i IN 100:
                IF Wind Farm Layout [i] == 1
                        APPEND [i mod 10, floor division(i/10)] to Turbines
                END IF
        END FOR
        Power = 0
        FOR TurbineA IN Turbines:
                Turbine Wind = 0
                TYPE LIST: Wind Speed Deficits
                FOR TurbineB in Turbines:
                        If TurbineA != TurbineB:
                                Turbine Angle = The angle between North, TurbineB and TurbineA
                                Wake Range = ArcTangent(Wake Decay Coefficient)
                                IF Turbine Angle is within the wind direction ± wake range:
                                        Distance = Downwind Distance between TurbineA and TurbineB
                                        APPEND Wind Speed Deficit to Wind Speed Deficits
                                END IF
                        END IF
                END FOR
                IF LENGTH of Wind Speed Deficits == 0:
                        Turbine Wind = wind speed
                ELSE IF:
                        FOR Wind Speed Deficit in Wind Speed Deficits:
                                Turbine Wind += Wind Speed Deficit^2
                        Turbine Wind = wind speed*(1-√Turbine Wind)
                END IF
                Power += 0.5*Air Density*Pi*Turbine Radius^2*Power Coefficient*Turbine Wind^3
        END FOR
        PASS OUT: Power
END FUNCTION
```

6.1 Using the final product

In order to run the code, you need four main files; main.py, wake_model.py, README.md and wind_data.txt. main.py directly accesses the other 3 files and is the main part of the code and is the file that is directly run in order to use the programme. wake_model.py contains subroutines Wake() and print_wind_farm() which are both used by main.py. wind_data.txt contains 1’044’711 data points of wind speed and wind direction, taken from La Haute Borne Wind farm [16]. README.md contains instructions on how to run the programme, including the external python modules that are needed; these are sklearn [10], pymoo [21], plotly [23], pandas [24] and numpy [25]. It also contains constants that can be changed by the user. These include the number of direction samples, the number of wind samples, the maximum wind speed sample,

the number of generations and population size in the Multi-Objective Optimisation algorithm, and the size of the wind farm grid (this is set to 10x10 by default and this remains consistent throughout the project. Increasing the wind direction and wind speed samples will increase the accuracy of the results at the cost of higher computational power required.

7 Analysis of the final product

7.1 Requirements

The main aim of this project is to use machine learning to optimise wind farm layouts. In order to do that, machine learning techniques were successfully used to create a Gaussian Mixture model to represent existing wind data in order to calculate future expected wind conditions. This was successfully implemented using SciKit Learn [10], as shown in 5.1. However, I achieved this using an external python module rather than coding it myself. Although it would have furthered my understanding of the process, it would have taken far more time and could have led to errors and mistakes in the final results. It would also likely have been more computationally expensive when coded myself as I cannot guarantee that my own implementation would be able to match the speed and efficiency of SciKit Learn [10]. This also applies to the Multi-Objective Optimisation function, which can be seen successfully implemented in 5.3 using Pymoo [21]. However, this is an argument against my implementation of the wake model, which I choose to code myself, and because the wake model and power output is calculated hundreds of thousands of times, it takes many hours to find optimal solutions. With more time I could have looked into optimising the efficiency of the wake model code. Despite this, the wake model still works and runs with no known errors, as seen in 5.2.

7.2 Results

With a population size of 400 and 800 generations, AGE/MOEA found 90 different wind farm layouts that lie within the Pareto front. For each 10 by 10 wind farm layout, an underscore means there is no turbine in that position and an 0 means that there is one in that position. Additionally, the first line of each layout is the northernmost line. For many layouts found with low numbers of wind turbines, the turbines are spaced out as expected, allowing the wake effect to decrease with distance. Something I did not expect is that for wind farm layouts with larger numbers of turbines (25+), the turbines start forming east-west rows of turbines next to each other. This can be explained by the fact that the wind direction in the dataset is not prominent at these angles, this can be seen most easily in Figure 10. This means that the wind conditions where the wake effect could cause a significant loss in power output by turbines being next to each other is rare.

Figure 19 shows some of the wind farm layouts that were found by AGE/MOEA. All 90 solutions in the pareto front are in the appendix.

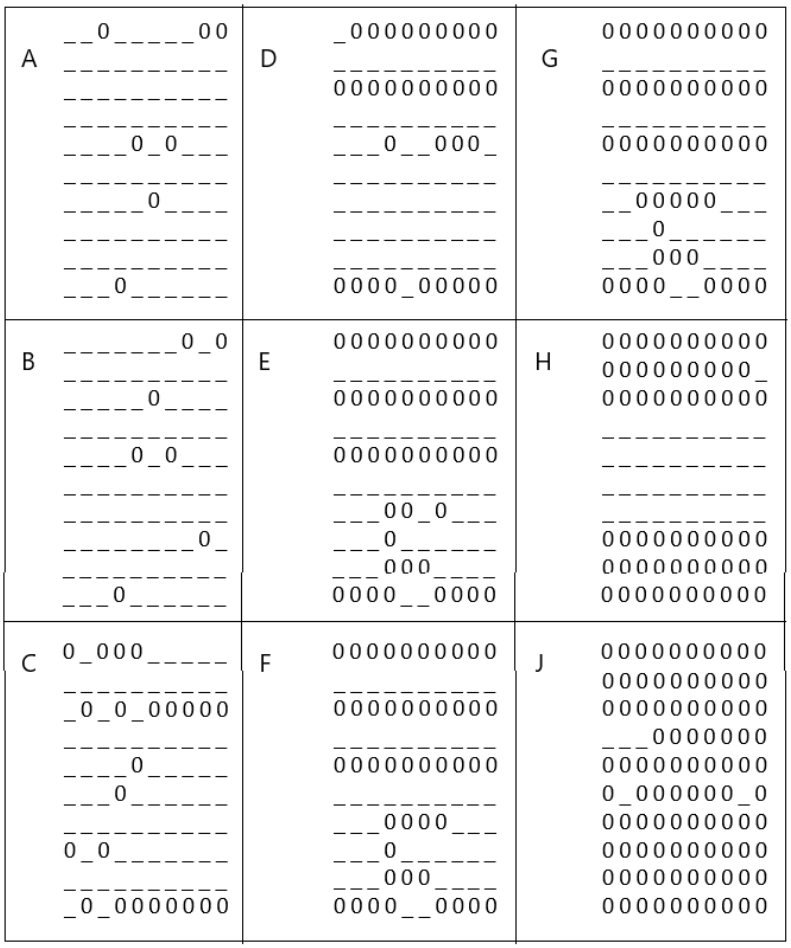


Figure 19: 9 of the 90 Solutions found by AGE/MOEA

Solutions A and B in Figure 19 shows the least expensive solution found using AGE/MOEA. Occasionally, some different turbine layouts produced the same cost and power output, solutions A and B are an example of that. Solution C is one of the first examples of rows starting to form with only 23 wind turbines. Solutions F and G have the same design as the solution that proceeds them but with just one more turbine added on. With 59 turbines, Solution H has a nearly identical symmetry between the north and south. Solution H has 95 turbines and is the most expensive solution found, which means that it is also the solution with the largest expected power output.

7.3 Performance and Accuracy

As mentioned in 7.1, it takes many hours to find solutions, which is because it is very computationally expensive, this is due to the amount of computation it takes to run the wake model hundreds of thousands, sometimes millions, of times. It can take many hours, or even days to run to completion depending on the constants set in the readme file. This is being run on my personal computer with a 3rd generation Ryzen processor [26], and a more powerful system would be able to run it faster.

In order to assess the accuracy, each component needs to be assessed. In 4.2, the Gaussian Mixture Model was found to have a silhouette score of 0.40, where 0 means there is no correlation between the model and the dataset and 1 means the Gaussian Mixture Model perfectly represents the dataset. Although a silhouette score of 0.40 is sufficient, it is definitely not perfectly accurate and a different model would potentially have represented the dataset better. Another way to make it more accurate is to increase the number of wind speed and wind direction samples, which is easily changeable in the readme file, but increasing either of these will also increase the runtime. The solutions found in 7.2 results were found using 18 wind direction samples for each 6 wind speed samples (a total of 108 wind condition tests per wind farm layout). As for the Jensen wake model, according to the paper in 3.2.1 [12], the Jensen wake model model is rather inaccurate for calculating wake losses under specific conditions, however it is considered to be reasonably accurate for predicting average energy losses over a large time scale, which is what this project's aim is. The final potential inaccuracy is some of the constants used in the wake model (air density, wake decay coefficient, power coefficient and thrust coefficient) are not constant and are dependent upon other conditions such as wind speed, altitude and air temperature, but for the sake of trying to focus on the main objectives and not overcomplicate the project, I found the most appropriate values for each of them and considered them as constants.

8 Critical assessment of the project as a whole

The main objective of this project is to use machine learning to optimise wind farm layouts under uncertainty. Ultimately, I have used SciKit Learn [10] (shown in 5.1) to implement a machine learning solution to eliminate some uncertainty created by differing wind conditions. And with the output of the project resulting in optimal wind farm layouts, and thus achieving the overall objective.

Python is a powerful and flexible language and was a good choice of programming language due to the vast library availability that I made use of. As discussed in sections 4.2 and 4.4, I relied heavily on the SciKit Learn [10] and pymoo [21] libraries. The main advantage of this is the efficiency and accuracy of these developed and maintained libraries, however, if I choose to implement them myself then I would have gained a more in depth understanding of these techniques and algorithms. I chose to implement the wake model myself and by doing so, the programme could have been more efficient if i had implemented an already existing library such as pywake [27], this could have reduced the long runtime. As a result, testing the multi-objective optimisation algorithm took an inconveniently long time, as it was the last part of the project that needed coding and required the other sections in order to be properly tested. I worked on the Gaussian Mixture model and wake model separately and thoroughly tested them each before starting the Multi-Objective Optimisation algorithm, this was in order to have confidence that the algorithm was working producing an appropriate and accurate output.

There are some potential improvements that could be made if I did this project again. The main improvement would be to try and spend more time finding different methods to better represent a wind dataset as the accuracy calculated by the silhouette score for the Gaussian Mixture Model in section 4.2 has room for improvement, maybe a single Gaussian model could have represented the data better than 2 Gaussian models. Another improvement that could

have been made is improving the wake model accuracy. This could have been achieved by using a different more accurate wake model or by trying to add more data to the wind conditions such as air pressure and temperature in order to recalculate some of the constants used and described in section 4.3. Alternatively, incorporating fluid dynamics and trying to simulate the wind and wake while taking into account the geography of the surrounding area could potentially be far more accurate but would be far more computationally expensive. The last major potential improvement would be to the resolution of the output; with more time and computational power, I could use different types of turbines, or different hub heights of turbines (such as paper 2.1.2 [7] did), or I could improve the square or rectangular grid that represents the wind farm. Instead, represent the wind farm with a more accurate representation of the target wind farm location, and maybe with more potential positions for turbines. Additionally, the programme could be coded to include a more diverse set of potential shapes for the designated wind farm area.

Overall the project was a success, and It has achieved the goals, adjectives and requirements. I feel I have learnt a great deal with machine learning trying to represent a dataset, using mathematical models to approximate the wake effect and using multi-objective optimisation algorithms in order to find a set of solutions to complicated problems.

9 Conclusion

This project is a practical method to help improve wind farms and subsequently increase renewable power output and global sustainability. It has provided a good insight into some challenges faced by wind farms and has provided a method and solution to one of those problems; optimising power output for the cost by utilising layout decisions. Overall, I have produced a program to create optimal wind farm layout under uncertainty. Using a given dataset, the program optimises the wind farm layouts for the area that the wind data was collected from. This means the results will not be optimised for a different location, however the method is repeatable with a different dataset.

Although similar research and systems are available, each method and different algorithm used will have its own pros and cons, and this project is tailored to a specific location. This means that any given wind farm would have difficulty optimising that location to the same extent as this project. Additionally, the wind data can be difficult to obtain. I got the wind data that was used for this project from a pre-existing wind farm [9], which demonstrated that this project is a proof of concept. Alternatively, with many wind turbines reaching the end of their operational lifetime [28], this project can serve as a way of using wind data that the pre-existing wind turbines gathered to produce a better layout for future wind farms that are constructed to replace old ones.

There are many different ways I could have approached this project and with more time, understanding and computational power, I could have made this project far more optimised, accurate and detailed; from incorporating fluid dynamics rather than using a model to estimate the wake effect, to trying to create more detailed and diverse options for the final wind farm layouts. With research and development into renewable energy, including wind power becoming more common and readily available, going into the future, I'm sure even more research and resources will go into other projects similar to this one in order to further optimise wind farms.

References

[1] https://www.nationalgeographic.com/environment/article/fossil-fuels (visited 18/04/22)

[2] Power Technology https://www.power-technology.com/features/featurethe-worlds-most-used-renewable-power-sources-4160168/ (visited 28/10/2021)

[3] earth.org https://earth.org/the-growth-of-renewable-energy-what-does-the-future-hold/ (visited 28/10/2021)

[4] WindFacts https://www.wind-energy-the-facts.org/wake-effect-7.html (visited 11/11/2021)

[5] Tuhfe Göçmen, Paul van der Laan, Pierre-Elouan Réthoré, Alfredo Peña Diaz, Gunner Chr. Larsen, Søren Ott. Wind turbine wake models developed at the technical university of Denmark: A review. 2015 https://www.sciencedirect.com/science/article/pii/S136403211600143X

[6] P. Mittal, K. Mitra. In search of flexible and robust wind farm layouts considering wind state uncertainty, Journal of Cleaner Production, 2020 https://www.sciencedirect.com/science/article/pii/S095965261934065X

[7] S.A. MirHassani, A. Yarahmadi. Wind farm layout optimization under uncertainty, Renewable Energy, 2017 https://www.sciencedirect.com/science/article/pii/S0960148117300733

[8] Zhang et al. Multivariate and Multimodal Wind Distribution Model Based on Kernel Density Estimation. ASME, 2011 https://www.researchgate.net/publication/229045319_Multivariate_and_Multimodal_Wind_Distribution_Model_Based_on_Kernel_Density_Estimation

[9] NDAWN https://ndawn.ndsu.nodak.edu/ (visited 08/11/2021)

[10] SciKit Learn https://scikit-learn.org/stable/modules/mixture.html (visited 09/11/2021)

[11] N.O. Jensen. A Note on Wind Generator Interaction. 1983 https://backend.orbit.dtu.dk/ws/portalfiles/portal/55857682/ris_m_2411.pdf

[12] Alfredo Peña, Pierre-Elouan Réthoré, M. Paul van der Laan. On the application of the Jensen wake model using a turbulence-dependent wake decay coefficient: the Sexbierum case. 2015 https://onlinelibrary.wiley.com/doi/full/10.1002/we.1863

[13] Andrew Kusiak, Haiyang Zheng, Zhe Song. On-line monitoring of power curves. 2008 https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.220.1552&rep=rep1&type=pdf

[14] Yusliza Yusoff, Mohd Salihin Ngadiman, Azlan Mohd Zain. Overview of NSGA-II for Optimizing Machining Process Parameters. 2011 https://www.sciencedirect.com/science/article/pii/S1877705811022466?via%3Dihub (visited 12/11/2021)

[15] Python 3.9.1 https://www.python.org/downloads/release/python-391/ (visited 14/04/22)

[16] Energie Renewables Opendata https://opendata-renewables.engie.com/explore/index (visited 06/11/2021)

[17] Air Density Calculator https://www.omnicalculator.com/physics/air-density (visited 16/04/22)

[18] Power coefficient http://xn--drmstrre-64ad.dk/wp-content/wind/miller/windpower%20web/en/tour/wres/cp.htm (visited 16/04/22)

[19] Python math library https://docs.python.org/3/library/math.html (visited 25/04/22)

[20] energypedia https://energypedia.info/wiki/Layout_of_Wind_Projects (visited 14/11/2021)

[21] Pymoo website https://pymoo.org/index.html (visited 13/11/2021)

[22] Feng Liu, Zhifang Wang. Offshore Wind Farm Layout Optimization Using Adapted Genetic Algorithm: A different perspective. 2014 https://www.researchgate.net/publication/261182428_Offshore_Wind_Farm_Layout_Optimization_Using_Adapted_Genetic_Algorithm_A_different_perspective

[23] Plotly https://plotly.com/ (visited 01/05/22)

[24] pandas https://pandas.pydata.org/ (visited 01/05/22)

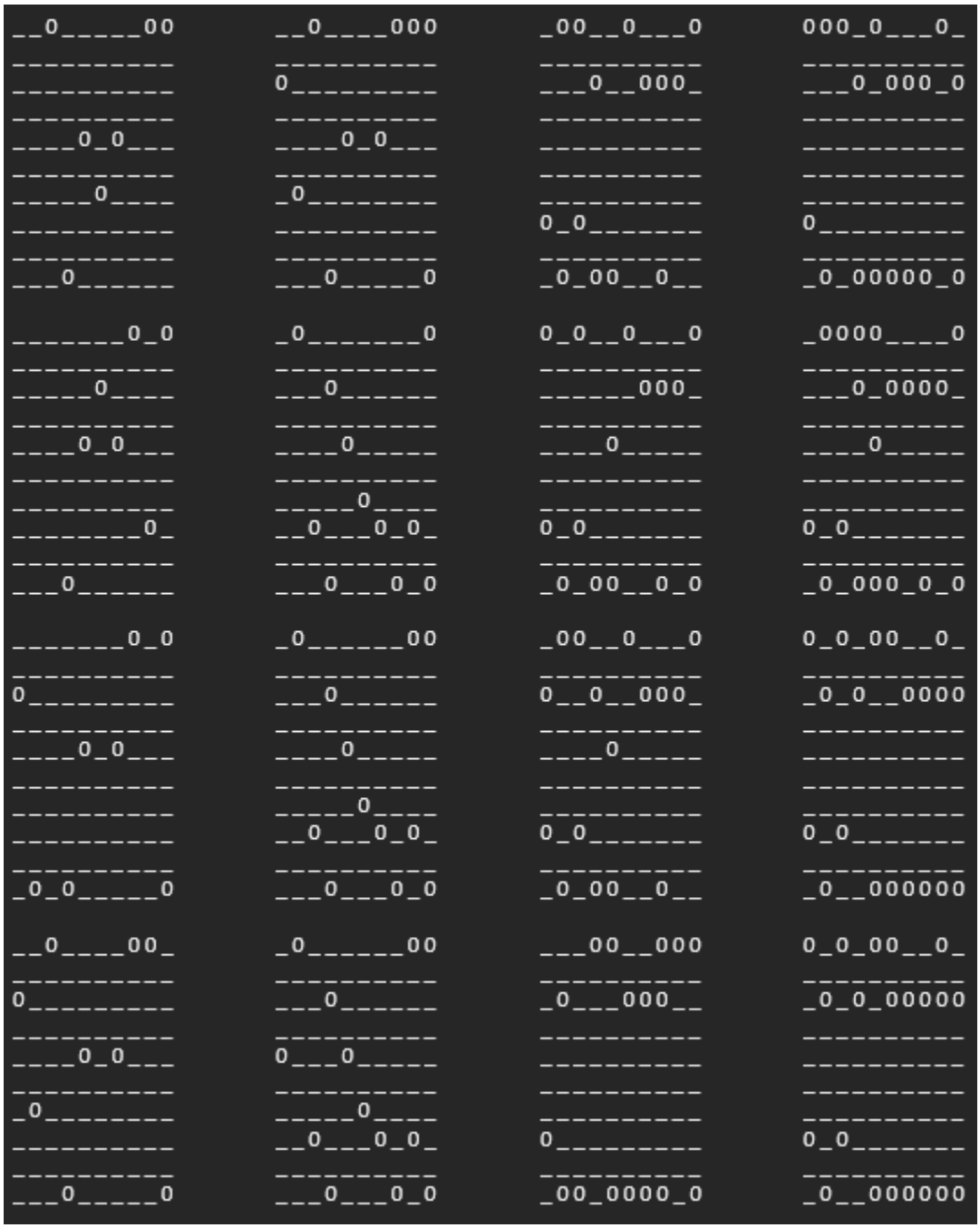
[25] NumPy https://numpy.org/ (visited 01/05/22)

[26] https://www.amd.com/en/products/cpu/amd-ryzen-5-3600 (visited 18/04/22)

[27] Pywake https://topfarm.pages.windenergy.dtu.dk/PyWake/notebooks/EngineeringWindFarmModels.html (visited 12/11/2021)

[28] https://windeurope.org/newsroom/press-releases/what-happens-when-wind-turbines-get-old-new-industry-guidance-document-for-dismantling-and-decommissioning/ (visited 24/04/22)

Appendix


__0_____00 __0____000 _00__0___0 000_0___0_
__________ __________ __________ __________
__________ 0_________ ___0__000_ ___0_000_0
__________ __________ __________ __________
____0_0___ ____0_0___ __________ __________
__________ __________ __________ __________
_____0____ _0________ __________ __________
__________ __________ 0_0_______ 0_________
__________ __________ __________ __________
___0______ ___0_____0 _0_00__0__ _0_00000_0
_______0_0 _0_______0 0_0__0___0 _0000____0
__________ __________ __________ __________
_____0____ ___0______ ______000_ ___0_0000_
__________ __________ __________ __________
____0_0___ ____0_____ ____0_____ ____0_____
__________ __________ __________ __________
__________ _____0____ __________ __________
________0_ __0___0_0_ 0_0_______ 0_0_______
__________ __________ __________ __________
___0______ ___0___0_0 _0_00__0_0 _0_000_0_0
_______0_0 _0______00 _00__0___0 0_0_00__0_
__________ __________ __________ __________
0_________ ___0______ 0__0__000_ _0_0__0000
__________ __________ __________ __________
____0_0___ ____0_____ ____0_____ __________
__________ __________ __________ __________
__________ _____0____ __________ __________
__________ __0___0_0_ 0_0_______ 0_0_______
__________ __________ __________ __________
_0_0_____0 ___0___0_0 _0_00__0__ _0__000000
__0____00_ _0______00 ___00__000 0_0_00__0_
__________ __________ __________ __________
0_________ ___0______ _0___000__ _0_0_00000
__________ __________ __________ __________
____0_0___ 0___0_____ __________ __________
__________ __________ __________ __________
_0________ _____0____ __________ __________
__________ __0___0_0_ 0_________ 0_0_______
__________ __________ __________ __________
___0_____0 ___0___0_0 _00_0000_0 _0__000000

```
0_000___00      000000____      _000000000      0000000000
__________      __________      __________      __________
_0_0_0000_      000_0_0000      0000000000      0000000000
__________      __________      __________      __________
__________      __________      ___0__000_      0_00000_0_
__________      __________      __________      __________
__________      __________      __________      __________
0_0_______      00_0______      __________      0_________
__________      __________      __________      __________
_0_0000000      0000000000      0000_00000      _000000000

0_000_____      0000000000      0000000000      0000000000
__________      __________      __________      __________
_0_0_00000      000000000_      000_000000      0000000000
__________      __________      __________      __________
____0_____      ______0___      ___00___00      0__00_0_0_
___0_______     __________      __________      __________
__________      __________      __________      __________
0_0_______      0_________      _0________      0_____0___
__________      __________      __________      _____0____
_0_0000000      _000__0000      _000000000      0000000000

00000_____      000000_000      0000000000      0000000000
__________      __________      __________      __________
_0___00000      00000_00__      000_000000      0000000000
__________      __________      __________      __________
__________      __________      ___00____0      0__0000_00
__________      __________      __________      __________
__________      __________      __________      __________
00______0_      00_____0__      00________      ______0___
__________      __________      __________      _____0____
0000000000      0000000000      0000000000      0000000000

00000__000      000000_0_0      0000000000      0000000000
__________      __________      __________      __________
000__000__      0000000000      000_000000      0000_00000
__________      __________      __________      __________
__________      ______0___      ___00____0      __000_0_0_
__________      __________      __________      __________
__________      __________      __________      __________
0______0__      00________      00____0___      0000__0_0_
__________      __________      __________      __________
_000000000      _000000000      0000000000      0000000000

00000__0__      0000000000      0000000000      0000000000
__________      __________      __________      __________
_0_0_00000      000_000000      0000000000      0000000000
__________      __________      __________      __________
__________      ______0___      0___000_0_      ___00_0_00
__________      __________      __________      __________
__________      __________      __________      __________
00______0_      00________      0_________      00_0__0___
__________      __________      __________      ____00____
0000000000      _000000000      0000000000      0000000000
```