Graphical Abstract

Mixed Integer Stochastic Optimization of relative Position of two Wind Turbines using using Neural Network based Constraint Learning

Simon Schmetz

Highlights

Mixed Integer Stochastic Optimization of relative Position of two Wind Turbines using using Neural Network based Constraint Learning

Simon Schmetz

- Research highlight 1
- Research highlight 2

Mixed Integer Stochastic Optimization of relative Position of two Wind Turbines using using Neural Network based Constraint Learning

Simon Schmetz

Universidad Carlos III de Madrid, test, test, test, test, test

Abstract

The following paper combines Linear Optimization with Constraint Learning to optimize wind turbine placement for maximum performance in a predefined area under randomly distributed wind. A Neural Network is trained on simulated data to model the impact of turbine positioning on power output. This model is integrated as a constraint in a linear optimization problem, and the problem evaluated for a current state-of-the-art wind farm configuration.

Keywords: Constraint Learning, Mixed Integer Optimization, Machine Learning, Neural Networks, Wind Farm Optimization

1. Introduction

With the clean energy transition currently taking place in Europe with ambitious targets for 2030 and beyond [1], wind energy is playing a central role in that transition and expected to rise to 50 % in the EU energy mix. [2] With wind energy thus expected to become one of the main contributers to the EU's energy production and large potentials identified for both onshore and offshore parks [3], attempts to optimize all parameters of windparks that result in even minor power efficeny improvement can be expected to yield significant returns in absolute power due to the scale of future wind energy production.

Within the main wind energy challanges lies the problem of optimizing the layout of wind farms. Here, the main goal is to reduce the negative impact that wake effects between wind turbines have on overall power generation, with yield reduction of up to 15% mainly due to reduced wind speeds in wake regions. Optimizing the farm for overall minimal wake exposure between wind turbines can thus potentially significantly increase the power output. [13] [14]

In practice, the problem reduces to placing wind turbines within a predefined zone, subject to the wind conditions as shown in Figure 1. These wind conditions can be assumed to be deterministic or (more accurately) random variables, by considering probability distributions for variables like wind direction and wind speed.

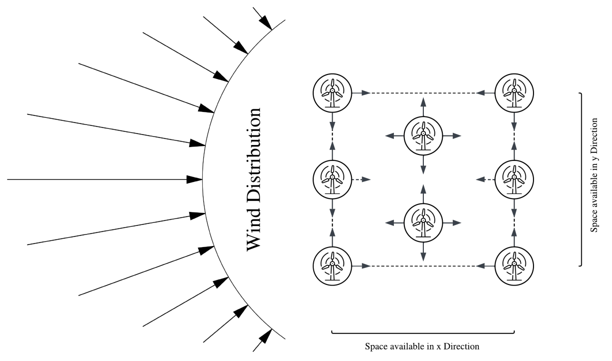

Wind Distribution
Space available in y Direction
Space available in x Direction

Figure 1: Optimizing the total power output of a farm reduces to placing wind turbines within a set space for the farm, subject to the wind conditions at the given location

This thesis is dedicated to a new approach for optimizing the placement of a fixed number of wind turbines in a predefined space, beginning with the two-turbine problem, the problem of optimally placing two turbines relative to each other. To solve this optimization problem, an extension to the Pyomo Python library is used, which allows the embedding of Neural Networks into the optimization problem as a set of constraints [5]. This extension allows the modelling of the effects of wind turbine placement relative to each other on power production. Introducing this model into the problem then allows for the optimization of overall power production across all wind turbines in the wind park.

To create a model optimally fit to the needs of the optimization problem, the model is trained on data specifically generated with the FLORIS [27] wind farm simulation tool for optimal coverage of the optimizations parameter space. To simplify the problem, the surface below the turbines is assumed to be perfectly flat and an equal wind speed is assumed along the entire height of the turbines. Solving the problem can be seperated into two main Steps:

1. *Farm Power Model:* Generation of simulation data covering the parameter space and training a Neural Network model with power generation as output
2. *Optimization:* Setting up optimization problem, embedding of power model and solving

This thesis is structured according to these two main steps, with a brief review of the state-of-the-art in wind turbine placement optimization and constraint learning beforehand.

2. Literature Review

Since the arrival of large-scale wind turbine farm operations as part of energy infrastructure, optimizing the positioning of the individual wind turbines relative to each other to mitigate wake effects and maximize the total power output is the subject of scientific investigation. As this Thesis is an attempt to apply a novel constraint learning method introduced in [5] to the optimization of wind farm layouts, the following state-of-the-art is split into two pieces: The first investigates the current relevant publications of wind farm optimization and the second one presents a brief introduction into recent developments in the field of constraint learning.

2.1. Optimization of Wind Farm Layouts

As discussed in the introduction, one of the main goals in the optimization of wind farm layouts is to reduce the negative impact of wake effects between wind turbines [14]. Historically, rule of thumb approaches were used by setting up the layout as a grid with the distance between wind turbines, with spacing in the dominant wind direction between 8 and 12 times the turbine rotor diameter and spacing perpendicular to the dominant wind direction 4 to 6 times the turbine rotor diameter [15] [13].

These methods have evolved to with pursuing the goal of maximizing the Annual Energy Production (AEP) of wind farms in the context of stochastic optimization as done in [25] [14].

The core of any of the most recent optimizations models is a wake model, which becomes part of the objective function to represent the wake effects on power output. These models can be categorized as [16]:

1. Experimental Methods
2. Numerical modeling
3. Analytical/semi-empirical modeling
4. Data-driven modeling

While experimental methods and numerical models might be the most precise models available for wake modeling, one of the challenges that come with the optimization is that the model has to be able to be introduced into the current state-of-the-art solvers as part of an objective function, leading to the prevalent use of analytical wake models like the Gaussian wake model and the 3D wake model [16]. With advancements in Machine Learning, the field of data-driven modeling is meanwhile expanding, with successful attempts in introducing Neural Networks and other Machine Learning frameworks into optimizations of wind farm layouts. Generally, either experimental data or data from numerical modeling (more prevalent) is used to train a chosen model type. The resulting model is then introduced into the optimization problem, as done in [17] [18] [19] [20].

2.2. Constraint Learning

The term Constraint Learning, defined as "finding a set of constraints, a constraint theory, that satisfies a given dataset" in [22], is the intersection of machine learning and optimization or more in practical terms, the introduction of machine learning models into optimization problems as constraints. As the in Machine Learnin the models learn from a given data set, the constraints resulting from such a model are equally learned. [22]

For this Thesis specifically, we consider the uncertainty aware constraint learning method of decomposing a neural network into a set of mixed-integer linear constraints [5] [21]. Similar approaches of embedding machine learning models into optimization problems have been taken for Decision Trees and Random Forests in [23] or for Neural Networks(without integer variables required), as done in [24]. A survey performed by Fajemisin et al.[6] shows how the field is currently emerging with an increase in publications in recent years and most publications revolving around the Embedding of Neural Networks and Decision Trees/Random Forests.

3. Farm Power Model

The first central component in optimizing the wind farm layout is to generate a data-driven surrogate Model that can be introduced into the optimization problem and solved by a solver. As detailed in the introduction, the goal is to use the distCL extension [30] to the Pyomo Python package, using a small Neural Network as a surrogate model. The following chapter documents the steps taken to generate such a model. To train a Neural Network, data is required that covers the parameter space of the optimization to prevent extrapolation by the model. Therefore, the chapter starts by explaining how the open-source wind farm simulation tool FLORIS [27] was used to generate a datase. Then, the fundamentals of Neural Network architecture and training are briefly introduced, before the pipeline that yields the final model is presented.

3.1. Data Source

The power output of wind turbines and wind farms as a whole is fundamentally connected to the aerodynamic conditions in the airflow every wind turbine experiences, with wind speed as the biggest factor relevant to how much power a wind turbine can generate. The main effect reducing the power generated by a wind turbine is to be positioned in the wake downwind of another turbine. This power reduction is primarily a result of the reduction in windspeed joined with the increased turbulence in the wake airflow [7]. Moving further downstream of a wind turbine, the wake gradually mixes with the outer airflow and thus again increases windspeed until the entire airflow reaches a new homogeneous air speed [8]. Thus, even if a wind turbine is positioned in the wake of another wind turbine, the greater the distance between the two, the less the second wind turbine is affected. Ideally, wind turbines are not positioned in the wake of other wind turbines at all.

With the overall goal of creating a model on how the interaction between wind turbines (induced by the wake) is related to the wind turbines relative positioning to each other and the wind conditions as input to an optimization model, a tailor-made dataset from simulation results is the easiest way to ensure that the data fits the use case well. After investigating two Python-based open source wind farm simulation tools FLORIS (FLORIS official website [28]) and PyWake (PyWake Documentation [29]), FLORIS was chosen due to its apparent ease of use.

As turbine type, an IEA 10-MW is used for data generation due to its repeated use as a baseline turbine in other scientific works like [10] and [11] and a more exact technical specification of the turbine can be found in [9]. The configurations of the specific dataset generated are chosen for the individual optimization problem formulations as detailed in Chapter 4.

3.2. Introduction to Neural Networks

To model the relationship between the attributes of an incoming airflow to a wind turbine and the output generated by the same wind turbine, many surrogate models could be chosen. As the model generated for this thesis is created to be then embedded into the Pyomo extension as explaind in Chapter 1, the model has to be compatible with said extension. That is why the model chosen is a simple neural network with a limited number of hidden layers and nodes, as large models increase the number of constraints required to decompose the model.

Neural Networks represent Graph Networks consisting of function blocks as the nodes, in the case of Neural Networks called "perceptrons" (or Neurons as a more general term) and arcs which correspond to the inputs/outputs of a given perceptron. In simple words, the inputs to the Neurons are summed up and introduced as an argument into a function $f()$, which yields an output y of the given perceptron. These Neurons are organized in layers, which correspond to the row-type structure Neural Networks are usually represented in and each perceptron of one layer is connected to every other perceptron of the following layer. The following layer, in this case, means in the direction of flow, in visualizations usually from left to right. The arcs thus correspond to a single number, and the Neurons to the action of summing up all inputs and then applying the unspecified function $f()$ to generate an output. This process is repeated for all Neurons in the network, with the first layer of the network taking as input the values of the given data features (the input values to the model) and the last layer producing outputs corresponding to the output value(s) of the model. The number of Neurons in this first and last layer corresponds to the task of the Neural Network. If the goal is to identify handwritten numbers 1-9 from 50x50 pixel images, the input layer might have 50x50 Neurons for the value of each pixel, and the output as 9 layers with the output value of each of those last perceptrons representing how much the network "thinks" the given image shows the corresponding numbers 1-9. A schematic of this architecture is given in Figure 2.

As shown in Figure 3 the output of a Neuron is the output y is generated by summing up the inputs x multiplied by a corresponding weight w together with a bias b and introducing this summation as an argument into an activation function $f()$. In this process, the weights w represent a weight to give importance to the individual inputs and the bias b serves to set a minimum output value that will always be reached, regardless of the inputs.

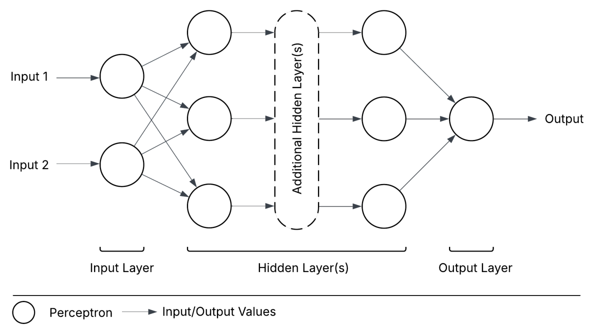


Figure 2: Schematic of Neural Network Architecture

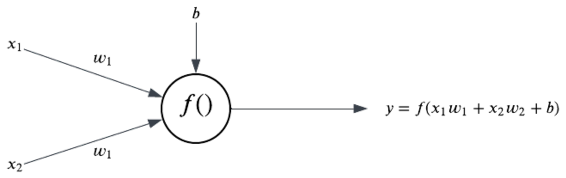


Figure 3: The output of a neuron is generated by applying the activation function to the sum as the weighted inputs $w_i x_i$ and the bias of the neuron b

The function $f()$ is called the activation function, as in its simplest form it represents a step function that decides if a neuron activates or not, e.g., takes the binary values $0, 1$ for a given threshold. Contrary to the human brain, where neurons are indeed binary, most Neural Networks resort to an activation function whose outputs are not binary but deliver continuous values between 0 and 1 to avoid the boundary issues that occur with thresholds. The most common function used in place of a step function is the sigmoid function, which roughly corresponds to a continuous version of the step function, with $\sigma(x) \approx 1$ for $x \to \infty$ and $\sigma(x) \approx 0$ for $x \to -\infty$.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

This relationship also becomes apparent when plotting both of those functions over each other, as shown in Figure 4. An alternative to the sigmoid as activation function is the Rectified Linear Unit (ReLu) function, or in simple words, the maximum function, defined as

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases}$$

The main general applicable benefit of the ReLu function is that it has a constant, non-vanishing gradient for both directions (see Figure 4), leading to better gradient propagation, an attribute that is relevant in the context of the backpropagation briefly discussed at the end of the Section. [26] Beyond this, the ReLu function allows for embedding the Neural Network into a optimization problem, as will be discussed in Section 4.2.

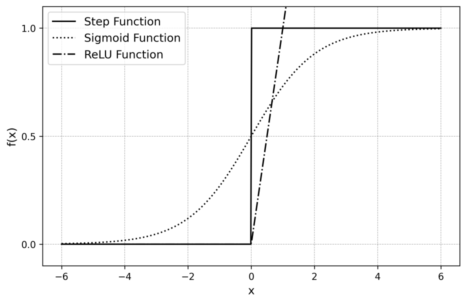


Figure 4: The Sigmoid Function being close to the Step Function without being as sensitive to slight changes in x due to its continuity. Alternatively, the Rectified Linear Unit function (ReLu) can be used as activation function.

[4]

The training of the Neural Network thus corresponds to adjusting the weights and biases in a way that minimizes the chosen loss function. The algorithm most commonly used for this is called *Backpropagation*. [4] The algorithm to do so corresponds to:

1. Introduce Training Data into the Neural Network
2. Evaluate the Loss Function for the Training Points (with small random values for weights and biases for the first iteration) and evaluate the gradients of the Loss function for the gradients of the individual weights and biases
3. Update the weights and biases according to a learning rate ρ weighted by the specific weight or biases gradient
4. Repeat process n Epochs, with one Epoch being the algorithm passing through all training points once

4. Stochastic Optimization and Constraint Learning

With the theoretical and pratical foundations of the model of farm/turbine power established, the following section treats the embedding of the trained model(s) into a deterministic/stochastic optimization problem. The underlying theory of optimization under uncertainty and constraint learning is introduced before the actual optimization problems for the two turbine optimization problem are defined, the required models trained, and the results discussed.

4.1. Optimization under Uncertainty

Like in many areas, knowledge about the uncertainty associated with certain variables can change decisions in which these variables underlying uncertainty plays a part. For example, while optimizing anything from next year's crop to tomorrow's energy pricing, the field of Stochastic Programming is occupied with finding methods that allow for introducing these uncertainties into optimization problems.

One way to approach these uncertainties is to split the problem into scenarios. A bakery, for example, has to decide on how many Baguettes to bake for the next day to maximize its profit. Baking too few Baguettes will lead to missing out on potential sales, while baking too many baguettes will mean that the demand is fulfilled, but the money invested in the excess number of baguettes is lost. Assuming the mean number of baguettes bought every day is 1000, the price to buy a baguette is 1 and the cost to produce a baguette is $0,2$ the classical approach to solving such a problem would be by the following formulation [1]:

$$\max_{x} \quad (1.00 \cdot \min(x, 1000) - 0.20 \cdot x)$$

To represent uncertainty, additional scenarios of the demand being 10% lower (900 Baguettes) and 10% higher (1100 Baguettes) can be added. Assuming that the mean demand has a 50% probability of occurring, the 10% demand increases a 20% probability and the 10% decrease a 30% probability, the problem can be modified to maximize the expected profit across these three scenarios by the formulation

[1] x non-negative

$$\begin{aligned}\max_{x} \quad & 0.5 \cdot (1.00 \cdot \min(x, 1000) - 0.20 \cdot x) \\ & + 0.3 \cdot (1.00 \cdot \min(x, 900) - 0.20 \cdot x) \\ & + 0.2 \cdot (1.00 \cdot \min(x, 1100) - 0.20 \cdot x)\end{aligned}$$

Assuming only these three scenarios are possible, the result from this optimization problem would be the Expected profit and how many Baguettes are the optimal number of Baguettes to yield the maximum Expected profit across all scenarios. This would be optimal assuming there is no more information about the next day's demand, meaning that by using this approach, the total profit over a long time would be maximal. In case there is more information regarding the next day's demand, the probabilities that give the weights in this optimization might shift, with one scenario potentially reaching probability 100% if there were to be absolute certainty that the next day's demand would be, for example, 1100 baguettes. As having such exact information is very rare, the best solution will be in most cases to maximize the profit Expectation.

The obvious connection to conventional statistical analysis is that the demand is a random variable that can take multiple values, in this case, we assumed it to be a discrete random variable Y with support $900, 1000, 1100$ even though in real-world applications the demand of baguettes will move somewhere between $[0, \infty]$. Finding the Expectation for such a discrete random variable can be done as

$$\mathbb{E}[X] = \sum_i x_i \cdot \mathbb{P}(X = x_i) = \sum_i x_i p_i$$

or for continuous random variables

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot f_X(x)\, dx$$

Using these two expressions, optimization problems can thus be formulated to optimize the Expectation of objective functions containing random variables. [12]

4.2. Constraint Learning

Constraint learning refers to introducing a model that has learned relationships between certain variables from data into an optimization problem.

In the case of constraint learning, the model is more specifically introduced into an optimization problem as a set of constraints. As many real-life relationships are dificult to represent by a simple analytical function, introducing machine learning models to optimization problems opens up many new possibilities [6].

In the case of Neural Networks, one way of introducing a Neural Network as a constraint into an optimization problem is by recognizing that when using the Rectifier Linear Unit (ReLu) Function as activation function with the (linear) sum of neuron bias and weighted inputs $\tilde{v}_i^\ell$ being the function argument

$$v_i^\ell = \max(0, \tilde{v}_i^\ell) = \max(0, b_i^\ell + \sum_j w_{ij}^\ell v_j^{\ell-1}) \tag{1}$$

the function can be rewritten as the following constraints

$$v_i^\ell \geq \tilde{v}_i^\ell \tag{2}$$
$$v_i^\ell \leq \tilde{v}_i^\ell - M^{\text{low}}(1 - j_i) \tag{3}$$
$$v_i^\ell \leq M^{\text{up}} j_i \tag{4}$$

with $j_i \in \{0, 1\}$ a integer variable such that

$$j_i = \begin{cases} 0 & \text{if } \tilde{v}_i^\ell < 0 \\ 1 & \text{if } \tilde{v}_i^\ell > 0 \end{cases} \tag{5}$$

This decomposition allows for introducing a Neural Network into an optimization problem by decomposing it into the shown set of mixed-integer linear constraints [5].

4.3. Quantile Discretization of Wind Direction Distribution

A problem that occours with the discrete scenarios introduced in Section 4.1 is that such scenarios like in the case of wind directions might leave large gaps in areas which in the continious distribution would experience a high probability density. This issue can be partially solved by discretizing not with a constant step length but based on the quantiles of the distribution. The discretization can be instead performed not by a constant length of the discretization steps, but by a constant probability within a chosen number of quantiles. For each quantile individually, the mean is then calculated and the resulting expected values for the quantile are taken as discretization steps. This approach, when used for the same Normal distribution with mean 270° and standard deviation 5°, fixing the scenario/quantile count to 7, yields the discretization shown in Figure 5.

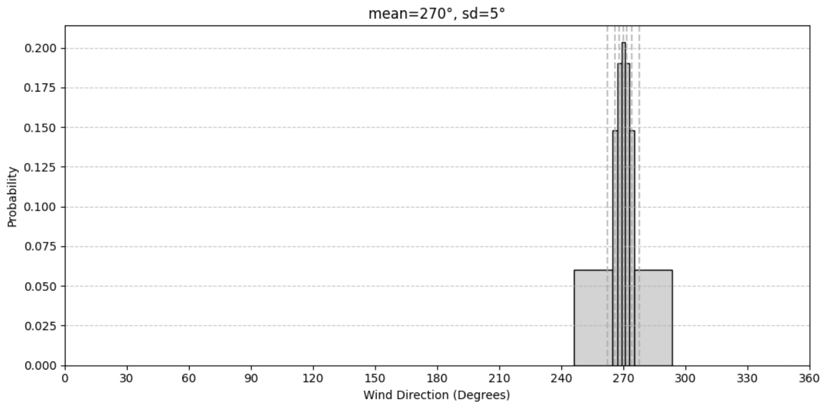


Figure 5: Quantile based discretization of Normal mean 270°, standard deviation 5°, disrcetized into 7 scenarios with equal probability for scenario and the scenario angle corresponding to the expected value for the quantile

Using this distribution, the expected power distribution changes for the corresponding scenarios/quantiles, with the space left between scenarios proportional to the probability density. When considering the previous approach to stochastic optimization, the number of scenarios that can be introduced into the problem has now become the main limitation. The following subsection provides an alternative set up that allows for a greater number of scenarios.

5. The Two Turbine Problem

Optimizing the positioning of two wind turbines can be expressed as optimizing the relative position of a second wind turbine T_2 to a fixed first turbine T_1, defined by the relative distances Δx and Δy. Both Δx and Δy are constrained by minimum distance Δ_{min} to T_2 and maximum distances $\Delta x_{max}/\Delta y_{max}$ to make the problem bounded. This problem can be visualized as shown in Figure 6.

The objective function to be optimized is the total power generation, e.g. the sum of power generated by both turbines. This objective is a function of

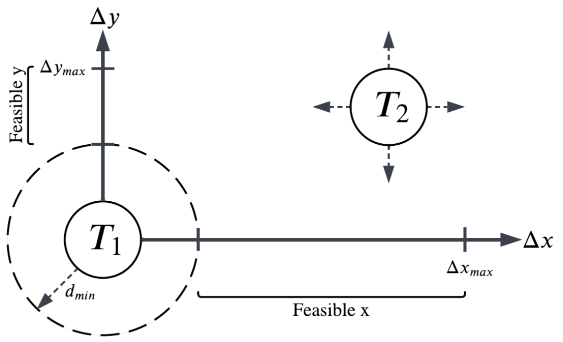


Figure 6: Optimizing the relative position $\Delta x/\Delta y$ of a second wind turbine T_2 to a fixed first turbine T_2, constrained by minimum distance d_{min} to T_1 and maximum distances $\Delta x_{max}/\Delta y_{max}$

both the position of the wind turbine as well as of the wind conditions like wind direction and wind speed.

$$f_{totalPower}(x, y, \text{windspeed}, \text{wind direction}, (...))$$

Different from the geographic coordinates, the wind condition parameters like windspeed are inherently not deterministic and follow distributions like the normal distribution as shown in Figure 7.

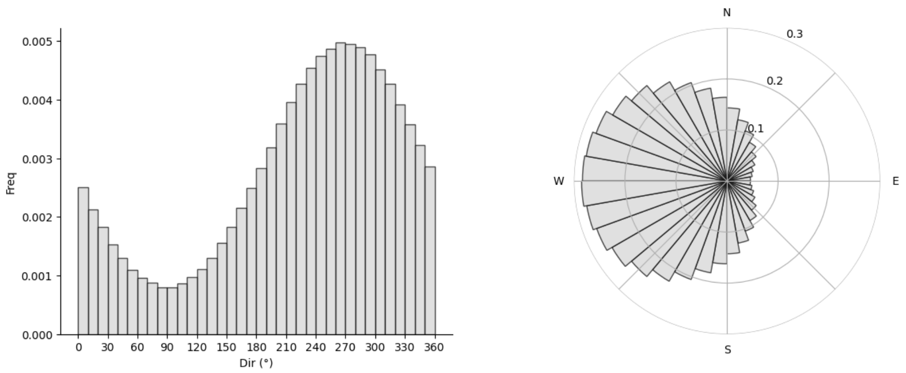


Figure 7: Histogram and Polar Plot across radiants for a normally distributed wind direction probability density function with mean West

The two main challenges of the optimization of the two-turbine problem are thus:

1. Introduce the complex relationship between turbine position, wind conditions, and power output into the optimization problem
2. Introduce the non-deterministic nature of wind conditions into the optimization problem

The first of these challenges is tackled by applying the Neural Network models discussed in Section 3.2 and introducing them into the optimization problem via constraint learning as described in Section 4.2. For the second problem, two approaches probabilistic approaches are explored in the following subsections.

5.1. Stochastic Optimization: Wind distribution indipendent Neural Network

The first stochastic formulation of the problem aims to set up the Neural Network in such a way, that makes it indipendent from the distribution of wind condition variables that is then used to generate scenarios for the calculation of the expectation. To do so, the objective function is a probability weighted sum, that represents the expected power generated by the farm. The Neural Network thus has to be able to predict the power output for a given turbine 2 location, taking into account the wind condition parameters (for example like direction and speed) at a given location. The problem is formulated as follows:

$$\max_{\mathbf{x},\mathbf{y}} \sum_{i=1}^{n} f_{Power,\mathrm{NN}}(\Delta x, \Delta y, \text{wind condition}) \cdot p_{n,\text{wind condtion combination}} \tag{6}$$

$$\text{s.t.} \quad \Delta x \leq X_{\max} \tag{7}$$

$$\Delta y \leq Y_{\max} \tag{8}$$

$$\sqrt{(\Delta x)^2 + (\Delta y)^2} \geq d_{\min} \tag{9}$$

Where:

- $(\Delta x, \Delta y)$ are the relative distances of the two turbines,
- $f_{Power,\mathrm{NN}}(\Delta x, \Delta y)$ is a deterministic neural network approximating the total power output for turbine positions and wind conditions
- $X_{\max}, Y_{\max}$ define the maximal distance the two turbines can be placed apart

- d_{min} is the minimum distance between the two turbines
- $p_{n,\text{wind condtion combination}}$ is the probability corresponding to each of the scenarios
- n is the index of the discretized possible combinations of wind conditions

The following section will first treat the univariate case of only wind direction being random, while wind speed and turbolence intensity remain constants.

The Notebook belonging to this formulation can be found in Stochastic optimization with farm power NN Fomulation Notebook [31]

5.1.1. Modelling

We begin by finding a fitting Neural Network model that can be embedded into the optimization problem. To do so, we again define the parameter space, this time with wind direction being variable and set up in a grid with step length of 10°. The full parameter grid is defined in Table 1

Table 1: Value Ranges for Probabilistic Two Turbine Problem Data Set

Variable	**Const/Variable**	**Value**	*Steplength*
Δx_{turb2}	Variable	[0, 5000] m	50 m
Δy_{turb2}	Variable	[0, 500] m	50 m
wind_speed	Constant	8 m/s	-
wind_direction	Variable	[180°, 270°]	10°
turbulence_intensity	Constant	0.06	-

We then proceed to hyperparameter tuning of the model. As apparent in Figure 8, the performance appears to flatline for models with or more than 2 layers and with or more than 50 nodes, yielding similar performance. To minimize the number of nodes, we thus proceed with the initial model configuration $\text{NN}(5 - 50^{\times 2} - 1)$ for the following optimization.

With a model chosen, we investigate its performance and find that while visually the shape of the wake generally appears to fit, the deviation in percent shows greater deviation, especially at greater distances away from Turbine 1 for the case of wind direction 260°, as shown in Figure 9a. The model appears to also show some artifacts, with the most notable being the

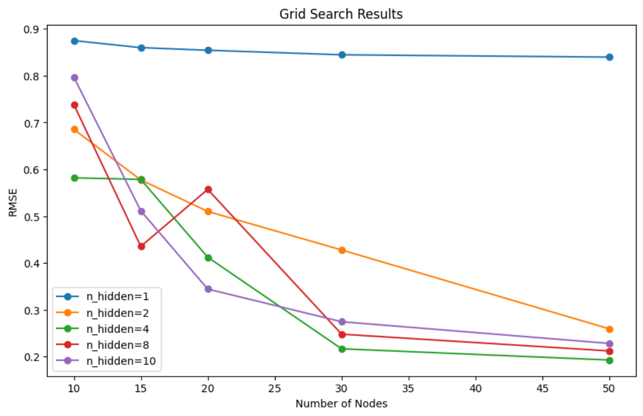
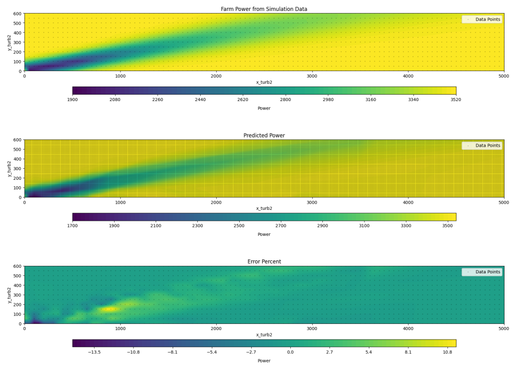


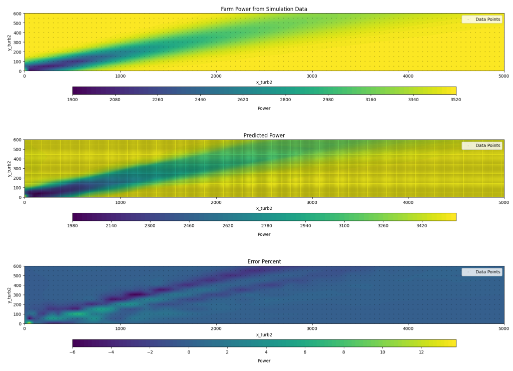
Figure 8: Grid Search results for a Neural Network configuration to predict for data with variable wind direction, showing similar performance for models of layers greater or equal than two, starting at 50 or more nodes

highest values in parallel to the wake right at the borders of the wake flow itself.

Having observed the deviation described above, another layer is added to the model in an attempt to prevent the previous behaviour. The resulting configuration NN$(5 - 50^{\times 3} - 1)$ improved the previous models weaknesses in that respect as can be seen in Figure 9b, which is why *the final model choosen for the optimization is NN*$(5 - 50^{\times 3} - 1)$



(a) Predictions of NN$(5 - 50^{\times 2} - 1)$



(b) Predictions of NN$(5 - 50^{\times 3} - 1)$

Figure 9: Comparison of model predictions: the left with two hidden layers, the right with three, showing reduced deviation in the final model.

While increasing the size of the Neural Network might further reduce the deviations across wind directions and for the individual wind directions, it is restricted by the solvability of the optimization problem. While there is no exact threshold, the solving time of the mixed integer problem that results from the embedding of the Neural Network increases roughly exponentially by adding Neurons to the Network and therefore adding binary variables to the problem. The shown model is deemed sufficient because of that trade-off.

For optimization purposes, the individual power per wind direction becomes less relevant, as the power outputs are weighted by their probability of occurring. To do so, first the probability distribution of the wind directions has to be defined, like the Normal distribution with mean 270° and standard deviation of 10° shown in Figure 10. To calculate the expectation for the retized parameter space using the trained model, this distribution itself has to be discretized in line with the initial parameter space defintion, which in this case is done in 10° steps.

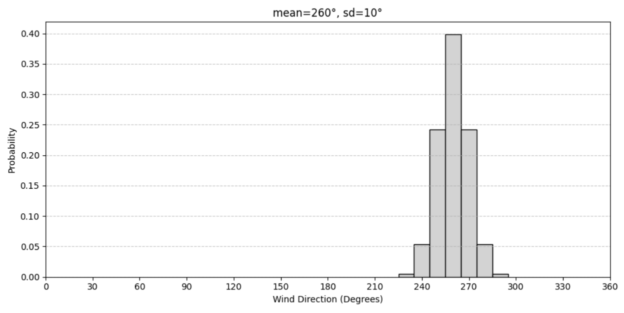


Figure 10: Discrete Wind Direction Probability Distribution with mean 270° and standard deviation 5°, discretized in intervals of 10°

Using this distribution, the expected power for the ranges of x and y coordinates can thus now be calculated by aggregating the powers for the different wind directions as a sum weighted by probability to yield the expected farm power generated at the specific coordinate. By evaluating the expected farm power both for the training data as well as for predictions for the training data, we can again compare the results and deviations as done in Figure 11. The directions with a probability greater than are shown as lines and their impact on the distribution of expected farm power visibel.

The normal distribution is used in this thesis as a placeholder for proof-of-concept purposes. In a practical application, the real wind distribution would have to be evaluated at the specific location of the wind farm.

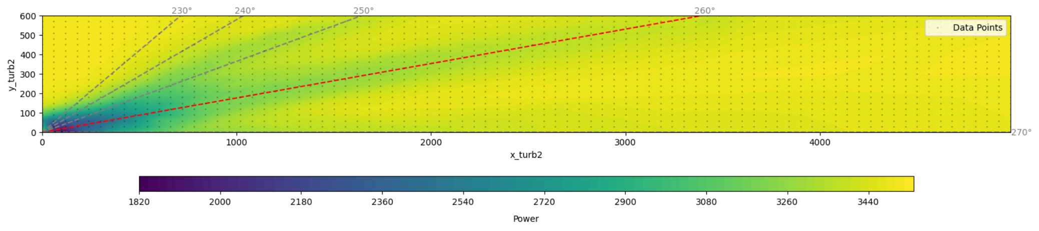


Figure 11: Distribution of Expectation of Farm power from a discretized Normal distribution with mean 260° and standard deviation 10° with steplength 10°

5.1.2. Optimization

With the model set up, solving the stochastic optimization problem for a maximum expected farm power output can be attempted. In the case of optimization, the problem is limited to three scenarios due to the computational constraints due to the size of the problem. A normal distribution with a mean of 270° and a standard deviation of 5° was therefore used as the wind direction distribution, yielding three scenarios (260°,270°,280°).

The Neural Network and the wind direction distribution are introduced into pyomo to find a global maximum. As previously, a range of optima exists, with Poyomo choosing among this area, as seen in Figure 12. More specifically in this case, the area close to the wake had been shown to predict higher values than in the simulations by the model (see Figure 9b), showing the optimization to find the global maximum within the provided Neural Network.

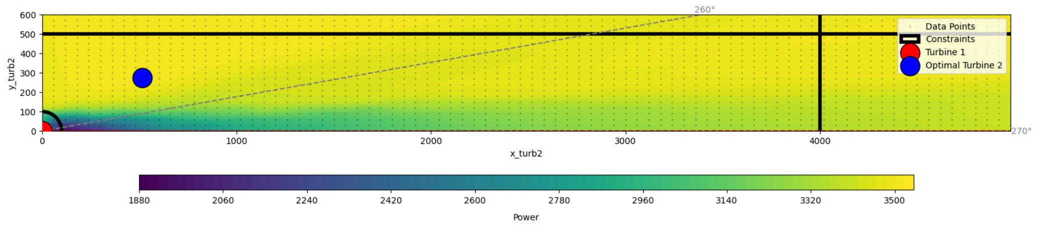


Figure 12

Like before, it is possible to visually find that there is a large range of more or less equally optimal points, with the few scenarios not covering the space appropriately. This leads to seemingly optimal areas in between the scenario wind directions, where in reality, significant wake losses can be expected. In addition to that, some small deviations of the model due to its complexity affect the optimization outcome significantly, with the current model predicting a maximum farm power production for turbine 2 placement close to the wake of turbine 1. Both of those aspects are weaknesses in the current configuration of the stochastic optimization.

5.2. Stochastic Optimization: Expectation Modelling Neural Network

While the expectation maximization has deemed to be the most accurate depiction of the problem on theoretical level, the previous method used for the resulting stochastic optimzation has shown to be severly limited by computational power, as a large Neural Network is required to model the farm power output both across x and y ranges of the second turbine, as well as across the wind condition variables. To overcome these constraints, a alternative formulation is set up, where the Neural Network is designed to output the expected farm power at any x/y point, given a certain wind condition distribution.

$$\max_{\mathbf{x},\mathbf{y}} \mathbb{E}[f_{Power}(\Delta x, \Delta y) \mid \text{wind condition distribution}]_{\mathrm{NN}} \tag{10}$$

$$\text{s.t.} \quad 0 \leq \Delta x \leq X_{\max} \tag{11}$$

$$0 \leq \Delta y \leq Y_{\max} \tag{12}$$

$$\sqrt{(\Delta x)^2 + (\Delta y)^2} \geq d_{\min} \tag{13}$$

Where:

- $(\Delta x, \Delta y)$ are the relative distances of the two turbines,
- $\mathbb{E}[f_{Power}(\Delta x, \Delta y) \mid \text{wind condition distribution}]_{\mathrm{NN}}$ a Neural Network that predicts the Expected Farm Power at al $(\Delta x, \Delta y)$ positions in the parameter space, given a specific wind speed distribution
- $X_{\max}, Y_{\max}$ define the maximal distance the two turbines can be placed apart
- $d_{\min}$ is the minimum distance between the two turbines

In practice, the new Neural Network configuration is achieved by training the Network directly on the Expected Farm Ppower Values, with the only remaining variables as the relative x/y position of the second wind turbine. The central benefit of this formulation is that the dimensionality of the Neural

Network is significantly reduced. The drawback ist that the model is thus condition on a spcific wind condition distribution and can not be used for any other distribution

The Notebook belonging to this formulation can be found in Stochastic optimization with farm power Expectation NN Fomulation Notebook [32]

5.2.1. Modelling

To generate the model with the farm power expectation as output, a more complicated process is required than in the previous formulation. For the Neural Network configuration used in this chapter, the Wind Condition distribution, which in this case will only be the distribution of wind directions, has to be defined. For the chosen distribution a discretization is generated the quantile based method described in Section 4.3. From this discretization, together with chosen limits for $\Delta x, \Delta y$, the parameter grid/space is defined and corresponding simulation performed to generate the data set. The probabilities from the wind condition distribution are then joined to the dataset and used to calculate the Expected Farm power for each $\Delta x, \Delta y$ combination in the parameter grid. This yields a significantly smaller data set with $\Delta x, \Delta y$ and Expected Farm power as features. This dataset is then used to train a model in the same way done previously, choosing a appropriat configuration of the model which then is used for embedding in the optimization problem. The described steps are shown in Figure 13.

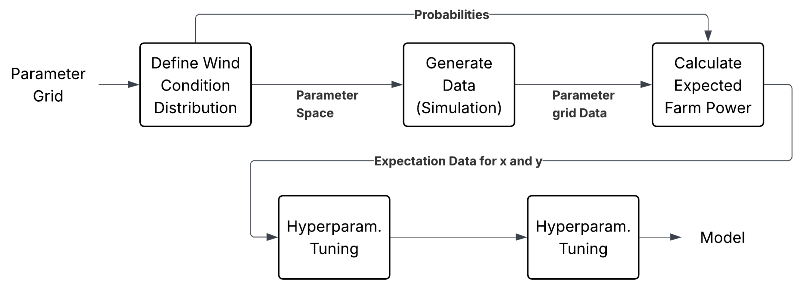

Probabilities
Parameter Grid
Define Wind Condition Distribution
Parameter Space
Generate Data (Simulation)
Parameter grid Data
Calculate Expected Farm Power
Expectation Data for x and y
Hyperparam. Tuning
Hyperparam. Tuning
Model

Figure 13: Flowchart displaying the steps used to set up a Neural Network model to model Expected Farm Power across Δx and Δy

As previously, we proceed to defining the parameter space, with the only difference compared to the previous stochastic formulation is that the wind direction how has variable steplength, using the method presented in Section 4.3.

Table 2: Value Ranges for Stochastic Expectation Neural Network Two Turbine Problem Data Set

Variable	**Const/Variable**	**Value**	*Steplength*
Δx_{turb2}	Variable	[0, 5000] m	50 m
Δy_{turb2}	Variable	[0, 500] m	50 m
wind_speed	Constant	8 m/s	-
wind_direction	Variable	[180°, 270°]	35 Quantiles
turbulence_intensity	Constant	0.06	-

As at this stage the parameter grid for wind direction is not yet well defined, the wind direction distribution and the number of discretization steps first has to be chosen. For this section, we proceed with the same normal distribution mith mean 260° and standard deviation 10° used in Section 5.1, but now discretized for 35 quantiles as shown in Figure 14.

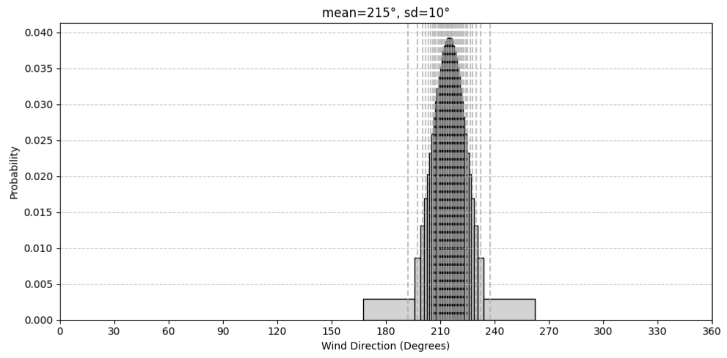


Figure 14: Quantile based discretized Normal with mean 260° and standard deviation 10°, discretized for 35 quantiles

With the parameter grid defined, the simulations can be performed, joint with the wind condition probabilities and expectations for the farm power calculated, yielding the dataset to be used in the training of the Neural Network. We thus proceed to the grid search optimization of the Neural Network configuration, as done in the previous chapters. For the distribution shown above, we find the 2-hidden layer and 20 neurons per hidden layer NN$(5-20^{\times 2}-1)$ configuration to yield the best results for least numbers of neurons, as can be seen in Figure 15.

We thus proceed with the configuration $NN(5 - 20^{\times 2} - 1)$.

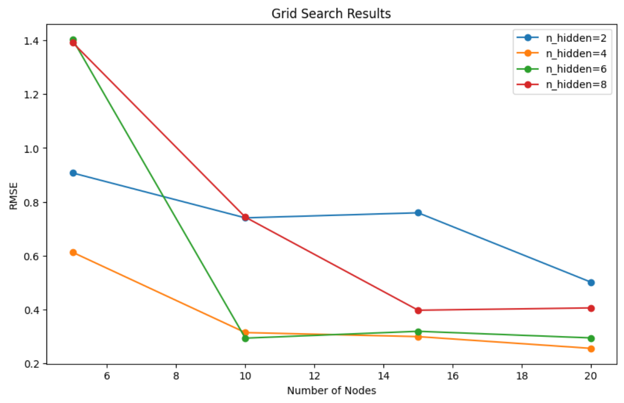


Figure 15: Grid Search Results for models trained on Farm Power Expectation data, with parameter grid as defined in Table 2 and Figure 14

The model of configuration $\mathrm{NN}(5 - 20^{\times 2} - 1)$ is thus trained and a brief visual inspection performed to investigate how well the model fits the data. As shown in Figure 16, does the model encorporate the shape of the Expected Power Distribution across x/y fairly well, with the only major deviations near the orgin, e.g. the position of Turbine 1 and directly after Turbine 1 in the principal wind direction.

The model thus appears to fit the physical behavior suficiently well to be able to proceed to the optimization step.

5.2.2. Optimization

The set up model is thus again embedded into a now seemingly deterministic optimization problem, as the uncertainty has already been taken into account when setting up the Neural Network itself through the expectations. The Neural Network is thus embedded directly into the problem, with the constraints as initially defined. The result shown in Figure 17 yield the expected result, with turbine 2 being placed as far upstream and as far away perpendicularly to the principal wind direction as possible from Wind Turbine 2.

Notably, the solving of this variation optimization problem requires significantly less computational recourses as the previous stochastic formulation, with a significantly larger quantity of scenarios considered for calculating the Expectation. In the entire optimization pipeline, the main limitation is

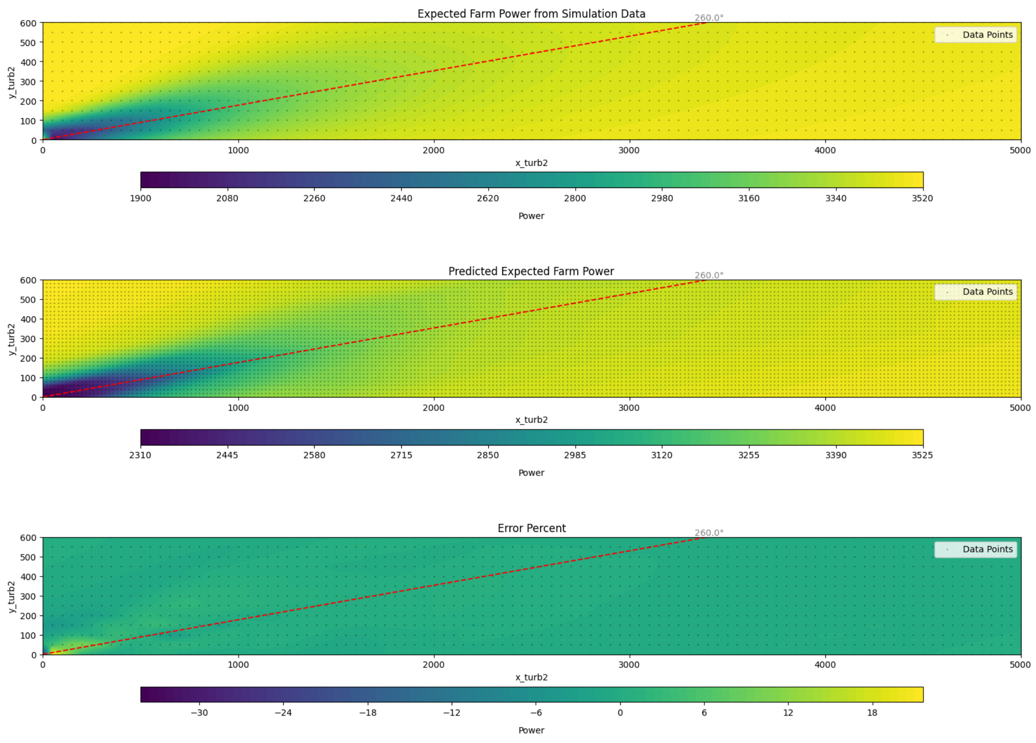


Figure 16: Visual comparison of training data vs predictions for Neural Network for Expectation of Farm Power, with linear interpolation in between data points to generate heatmap

in this formulation constitued by running the simulations required for the individual discretization of different wind direction/wind condition distributions. Due to the efficency of the FLORIS simulations, this does not seem to be a significant limitation for the two turbine configuration at this moment however. Therefore, this method appears to be better fit to scale up the complexity of the wind condition distribution, allowing for expanding to a multivariate distribution of the potentially dependent variables wind direction and wind speed for example.

6. Conclusion

The Paper contains the training of a Neural Network to predict the total farm power/expected farm power generated by a wind farm consisting of two turbines, with the inputs to the model as the relative position of the second

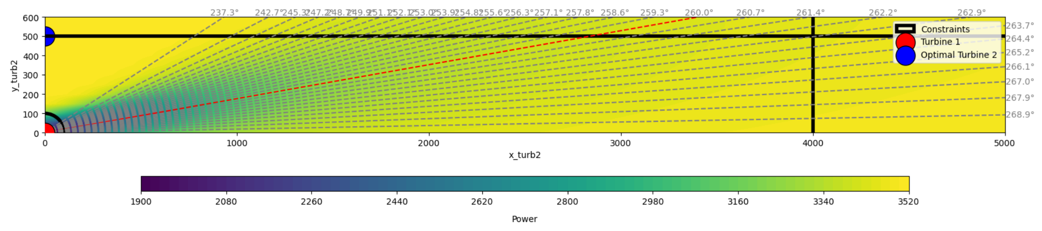


Figure 17: Optimization Result for maximization of Expected Farm Power, with wind direction Normally distributed with mean 260° and standard deviation 10°, discretized into 35 scenarios using quantile based discretization and Neural Network configuration $\mathrm{NN}(5-20^{\times 2}-1)$

wind turbine to the first as well as the wind condition parameters like wind direction. The resulting Neural Networks where embedded into a mixed integer linear programming problem using an approach for decomposing a Neural Network into lmixed-integer inear constraints. Two optimization problems where solved using this aproach, for a stochastic optimizations with a scenario based approach, to maximize the expected power generation, subject to a discreitzed distribution of the wind direction. Here two variants where developed, one via a large general neural network with the wind condition parameters as additional variables and one via a reduced Neural Network, trained on the already evaluated expected farm power for each x/y position of the second turbine.

The first formulation, using a larger Neural Network indipendent of the wind condition distribution yielded a versatile variant, with a Neural Network that could be applied to any location and any wind condition distribution, while only being able to consider a low number of scenarios due to computational constraints. The second stochastic formulation evaded the computational limitations by training its model directly on the farm power expectations, making both the Neural Network and the optimization model significantly more efficient, but loosing generality due to the Neural Networks conditioning on the wind condistion distribution.

Overall, the second stochastic formulation appears to be the only variant that sufficiently considers the randomness of the wind conditions like wind speed, while showing a path to scaling up computationally.

With the second stochastic formulation having shown to deliver good results, the next steps would revolve around investigating how to generalize the formulation to a larger number of wind turbines. Here, a fundamental reformulation might be required as the current formulation using relative positions would quickly grow the parameter space exponentially.

None the less it might be worthwile to pursue as the underlying Constraint Learning Methodology has been shown to be able to cope with the complexity of wind turbine wakes in this Thesis.

References

[1] European Commission, *Renewable energy targets*, `https://energy.ec.europa.eu/topics/renewable-energy/renewable-energy-directive-targets-and-rules/renewable-energy-targets_en`, 2023. Accessed: 2025-04-03.

[2] Analysis and Research Team, General Secretariat of the Council of the European Union, *Harnessing Wind Power: Navigating the EU Energy Transition and Its Challenges*, `https://www.consilium.europa.eu/media/1kyk0wjm/2024_685_art_windpower_web.pdf`, September 2024. Accessed: 2025-04-03.

[3] European Environment Agency, *Europe's Onshore and Offshore Wind Energy Potential*, Technical report 6/2009, `https://www.eea.europa.eu/en/analysis/publications/europes-onshore-and-offshore-wind-energy-potential`, 2009. Accessed: 2025-04-03.

[4] Michael Nielsen, *Neural Networks and Deep Learning - Chapter 1*, `http://neuralnetworksanddeeplearning.com/chap1.html`, 2015. Accessed: 2025-04-10.

[5] Antonio Alcántara and Carlos Ruiz, *A neural network-based distributional constraint learning methodology for mixed-integer stochastic optimization*, Expert Systems with Applications, Volume 232, 2023.

[6] Adejuyigbe O. Fajemisin and Donato Maragno and Dick den Hertog, *Optimization with constraint learning: A framework and survey*, European Journal of Operational Research, Volume 314, Number 1, Pages 1-14, 2024.

[7] C.T. Kiranoudis and Z.B. Maroulis, *Effective short-cut modelling of wind park efficiency*, Renewable Energy, Volume 11, Number 4, Pages 439-457, 1997.

[8] M. Magnusson and A.-S. Smedman, *Air flow behind wind turbines*, Journal of Wind Engineering and Industrial Aerodynamics, Volume 80, Number 1, Pages 169-189, 1999.

[9] Pietro Bortolotti and Helena Canet Tarrés and Katherine Dykes and Karl Merz and Latha Sethuraman and David Verelst and Frederik Zahle, *IEA Wind Task 37 on Systems Engineering in Wind Energy - WP2.1 Reference Wind Turbines*, National Renewable Energy Laboratory, 2019.

[10] Mads Madsen and Frederik Zahle and Sergio Gonzalez Horcas and Thanasis Barlas and Niels Sørensen, *CFD-based curved tip shape design for wind turbine blades*, Wind Energy Science, Volume 7, Pages 1471-1501, 2022.

[11] Samuel Kainz and Julian Quick and Mauricio Souza de Alencar and Sebastian Sanchez Perez Moreno and Katherine Dykes and Christopher Bay and Michiel Zaaijer and Pietro Bortolotti, *IEA Wind TCP Task 55: The IEA Wind 740-10-MW Reference Offshore Wind Plants*, National Renewable Energy Laboratory, NREL/TP-5000-87923, 2024.

[12] John R. Birge and François Louveaux, *Introduction to Stochastic Programming*, Springer Series in Operations Research and Financial Engineering, Springer, 1997.

[13] Peng Hou and Jiangsheng Zhu and Kuichao Ma and Guangya Yang and Weihao Hu and Zhe Chen, *A review of offshore wind farm layout optimization and electrical system design methods*, Journal of Modern Power Systems and Clean Energy, Volume 7, Number 5, Pages 975-986, 2019.

[14] Taewan Kim and Jeonghwan Song and Donghyun You, *Optimization of a wind farm layout to mitigate the wind power intermittency*, Applied Energy, Volume 367, 2024.

[15] F. Azlan and J.C. Kurnia and B.T. Tan and M.-Z. Ismadi, *Review on optimisation methods of wind farm array under three classical wind condition problems*, Renewable and Sustainable Energy Reviews, Volume 135, 2021.

[16] Li Wang and Mi Dong and Jian Yang and Lei Wang and Sifan Chen and Neven Duić and Young Hoon Joo and Dongran Song, *Wind turbine wakes modeling and applications: Past, present, and future*, Ocean Engineering, Volume 309, 2024.

[17] Kun Yang and Xiaowei Deng and Zilong Ti and Shanghui Yang and Senbin Huang and Yuhang Wang, *A data-driven layout optimization framework of large-scale wind farms based on machine learning*, Renewable Energy, Volume 218, 2023.

[18] N. Bempedelis and F. Gori and A. Wynn and S. Laizet and L. Magri, *Data-driven optimisation of wind farm layout and wake steering with large-eddy simulations*, Wind Energy Science, Volume 9, Number 4, Pages 869-882, 2024.

[19] Zilong Ti and Xiao Wei Deng and Hongxing Yang, *Wake modeling of wind turbines using machine learning*, Applied Energy, Volume 257, 2020.

[20] Zilong Ti and Xiao Wei Deng and Mingming Zhang, *Artificial Neural Networks based wake model for power prediction of wind farm*, Renewable Energy, Volume 172, Pages 618-631, 2021.

[21] Antonio Alcántara and Carlos Ruiz and Calvin Tsay, *A quantile neural network framework for two-stage stochastic optimization*, Expert Systems with Applications, 2025.

[22] Luc De Raedt and Andrea Passerini and Stefano Teso, *Learning constraints from examples*, Proceedings of the AAAI conference on artificial intelligence, Volume 32, 2018.

[23] Alessio Bonfietti and Michele Lombardi and Michela Milano, *Embedding Decision Trees and Random Forests in Constraint Programming*, 2015.

[24] Héctor G. -de-Alba and Andres Tellez and Cipriano Santos and Emmanuel Gómez, *A reformulation to Embedding a Neural Network in a linear program without integer variables*, arXiv:2402.02086, 2024.

[25] Michael Sinner and Paul Fleming, *Robust wind farm layout optimization*, Journal of Physics: Conference Series, Volume 2767, Number 3, 2024.

[26] Xavier Glorot and Antoine Bordes and Y. Bengio, *Deep Sparse Rectifier Neural Networks*, Journal of Machine Learning Research, Volume 15, 2010.

[27] National Renewable Energy Laboratory (NREL), *FLORIS: FLOw Redirection and Induction in Steady State*, GitHub repository, Version 4.4.2, `https://github.com/NREL/floris`, 2025.

[28] National Renewable Energy Laboratory (NREL), *FLORIS: FLOw Redirection and Induction in Steady State*, `https://www.nrel.gov/wind/floris`, March 2025.

[29] Mads M. Pedersen and Paul van der Laan and Mikkel Friis-Møller and Alexander Meyer Forsting and Riccardo Riva and Leonardo Andrés Alcayaga Román and Javier Criado Risco and Julian Quick and Jens Peter Schøler Christiansen and Bjarke Tobias Olsen and Rafael Valotta Rodrigues and Pierre-Elouan Réthoré, *PyWake: an open-source Python wind farm simulation tool*, `https://topfarm.pages.windenergy.dtu.dk/PyWake/`, 2025.

[30] Antonio Alcantara and Carlos Ruiz, *DistCL: A Neural Network-Based Distributional Constraint Learning tool for Mixed-Integer Stochastic Optimization*, `https://github.com/antonioalcantaramata/DistCL`, 2022. Accessed: 2025-06-06.

[31] Simon Schmetz, *Two-Turbine Problem with Constraint Learning and Probability-Weighted Objective*, `https://github.com/schmeti/uc3m_TFM_wind_farm_optimization_codebase/blob/main/Windfarm_power_modelling/0_two_turbine_problem_constrLearn_probweighted.ipynb`, 2025. Accessed: 2025-06-06.

[32] Simon Schmetz, *Two-Turbine Problem with Constraint Learning and Probability-Weighted Objective (Expectation Neural Network)*, `https://github.com/schmeti/uc3m_TFM_wind_farm_optimization_codebase/blob/main/Windfarm_power_modelling/0_two_turbine_problem_constrLearn_probweightedt_expNN.ipynb`, 2025. Accessed: 2025-06-06.