Optimising Wind Farm Layout for Maximum Energy Production

Aayush Shah, Amal Thomas, K S Varun, Rahul Patel, Vinayak Gautam

Abstract - Comparative analysis of Particle Swarm Optimisation and Pattern Search algorithm is conducted for the problem of wind farm optimisation. The optimisation objective is to maximise the energy production of a wind farm with a fixed number of turbines, given constraints on land area and inter-turbine spacing.

1 Introduction

With the growing consensus to turn towards renewable energy sources to meet society's current energy requirements, wind farms have emerged as a viable option for energy generation. Wind farms are groups of wind turbines that are strategically placed in specific locations to generate electricity. The factors affecting the power generation from such farms are wind speed, turbine design, windmill layout, etc.
Wind turbines in a wind farm typically exhibit inferior performance when compared to isolated turbines that operate under free-stream wind conditions. This is because the wakes generated by turbines can limit the amount of energy available to downstream turbines. Consequently, if a wind farm is poorly designed, there may be significant reduction in its power generation capacity. In this project, we conduct a comparative analysis of two optimisation algorithms to determine the most efficient windmill layout for maximizing energy production, given a fixed area of land and a fixed number of turbines. These algorithms are:

- Particle Swarm Optimisation
- Pattern Search Algorithm

It was found that these algorithms often get stuck in local maxima of the function representing power production. To overcome this, we develop a "Greedy" algorithm to discover improved initial configurations, which when fed into the above algorithms, increase the chances of finding the global maximum.

2 Modelling a single wind turbine

Assumptions:

- Wind flow occurs in axial direction in and out of the control volume
- Ideal rotor (thin, frictionless)
- No change in the potential energy of air stream

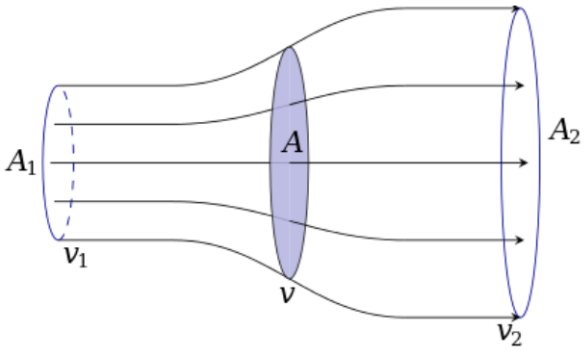

Figure 1: Schematic of fluid flow through a disk-shaped actuator

In our streamtube, we have the fluid flowing from left to right, and an actuator disk that represents the rotor. The fluid velocity slows down from v_1 to $v = v_1(1-a)$ at the rotor. We can obtain an expression for the pressure difference between the two sides of the rotor by applying Bernoulli's equation:

$$P_{D+} - P_{D-} = \frac{1}{2}\rho(v_\infty^2 - v_w^2)$$

Force acting on the actuator disk can be calculated as:

$$F = \Delta P A$$

The power extracted from the fluid is the force acting on the fluid multiplied by the velocity of the fluid at the point of power extraction:

$$\text{Power}_{ext} = F v_D = 2a(1-a)^2 v_\infty^3 \rho A D$$

We are interested in finding the maximum power that can be extracted from the fluid. Differentiating the above equation with respect to 'a' and equating it to zero, we get $a = 1/3$. Using this, we get the maximum fraction of power that can be extracted from the fluid (theoretical limit) as:

$$C_{P_{max}} = 16/27$$

Therefore, in our model, we set $a = 1/3$. However, we simulate each wind turbine to have a **power coefficient** (C_P) of only $0.8 \times C_{P_{max}}$, to account for practical inefficiencies. The wind velocity at the end of the streamtube is calculated as $v_\infty(1-2a) = v_\infty/3$. After this, the wind streamtube gradually expands to once again achieve the ambient wind velocity v_∞. This region is termed as the **wake region**.
The simplest model that best encapsulates this, is the **Jensen Model** [2], which represents the velocity of air in the wake region using the following equation (α is the wake expansion coefficient, set to 0.1):

$$v = V_{inf}(1 - \frac{2}{3}(\frac{D/2}{D/2 + \alpha x})^2)$$

We will use the above model as the basis for the simulation of our wind farm.

3 Mathematical Formulation

Nomenclature
S : Length on one side of the square plot of land
N : Number of turbines

ρ : Density of air
D : Diameter of the turbine i.e., twice the blade length
A : Cross section area of turbine
V_∞ : Ambient wind velocity

$$\textbf{Decision variables}: x_i, y_i \quad \forall \quad i \in \{1, 2, ..., N\}$$

$$\textbf{Objective Function}: \text{Power} = \frac{1}{2} C_p \rho A \Sigma_{i=1}^{N} V_{eff,i}^3$$

Maximize Power, subject to: $0 \leq x_i < S$, $0 \leq y_i < S$,

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq 2D \quad \forall \quad i, j \in \{1, 2, ..., N\}, i \neq j$$

$$V_{ij} = \begin{cases} V_{eff,j}\left(1 - \frac{2}{3}\left(\frac{D/2}{D/2 + \alpha(x_j - x_i)}\right)^2\right) & \text{, if } \; y_j - \alpha(x_i - x_j) - D/2 < y_i < y_j + \alpha(x_i - x_j) + D/2 \\ V_{eff,j} & \text{, otherwise} \end{cases}$$

where, V_{ij} is the wake velocity exerted by the turbine at (x_j, y_j) at the location (x_i, y_i)

Multi-turbine wake velocity superposition model - Velocity Geometric Sum (V-GS) formula :

$$V_{eff,i} = V_{inf} \Pi_{j=1}^{N} \frac{V_{ij}}{V_{eff,j}}$$

4 Approach

4.1 Particle Swarm Optimisation

Particle Swarm Optimisation is an optimisation technique inspired by the biological phenomenon of a flock of birds looking for food. A number of candidates for the optimal solution, called particles, are initialized which explore around for locally optimum solutions. A main feature of this algorithm is communication between the particles. Any particle which has identified the best solution yet will convey it to other particles, so that they gravitate towards that location.

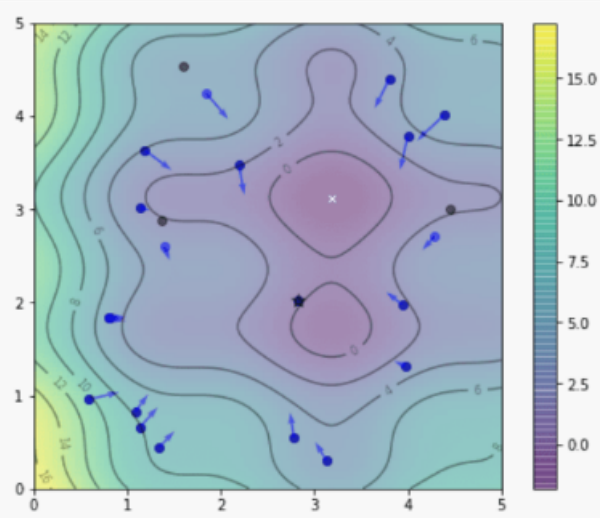

Figure 2: Particles in PSO (Only for representational purposes)

In the context of our problem, one particle defines the entire wind farm configuration, hence the particle here is 60 dimensional, containing x and y coordinates of each of the 30 wind turbines (taking, N = 30).

$$P_i^{t+1} = P_i^t + V_i^{t+1}$$

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 (P_{best(i)}^t - P_i^t) + c_2 r_2 (P_{bestglobal}^t - P_i^t)$$

Here, Xi refers to a single particle, Vi is its velocity which is defined by the update rule as given. w, c1 and c2 are the parameters to the algorithm.
Exploitation is the ability of particles to target the best solutions found so far. Exploration, on the other hand, is the ability of particles to evaluate the entire research space. The challenge of the remaining part of the article will be to determine the impact of these coefficients to find a good balance between exploration and exploitation.
In the context of our problem, one particle defines the entire wind farm configuration, hence the particle here is 60 dimensional, containing x and y coordinates of each of the 30 wind turbines.

4.1.1 Greedy Initial Configuration

We have defined a Python function windfarm() which takes the arguments N, S and D to generate an initial configuration to be used for Particle Swarm Optimisation. The function greedily chooses a point (x_i, y_i) that has the maximum velocity one by one, subject to the distance constraint:

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq 2D \quad \forall \quad i, j \in \{1, 2, ..., N\}, i \neq j$$

This method involves 2 matrices, territory matrix and velocity matrix (size S $\times$ S), both with all values initialized to 1.
The territory matrix is a binary matrix, i.e. its values are either 0 or 1. Each element of the matrix represents a coordinate on the wind farm. A turbine can only be placed on a certain (x, y) coordinate when its corresponding value in the territory matrix is 1.
The velocity matrix stores the ratio of wind velocity to the ambient wind velocity for all integer coordinate values in the wind farm. If the wind farm is empty, all values in the velocity matrix are equal to 1. In the presence of wind turbines, some regions experience lower wind velocity due to turbulence and these points will have the ratio to be less than 1.

This function is greedy since it always chooses the point which has the maximum velocity ratio associated with it, to place a turbine. In order to satisfy the distance constraint, the territory matrix and velocity matrix are multiplied together element-wise, and the point (x_i, y_i) is chosen from this resulting matrix.
To account for multiple points with the same maximum velocity, priority is given to the points as follows:

- Higher priority is given to the point with the lowest X coordinate value since the power generated by a turbine is only affected by the turbines to the left of it or ahead of it.
- The point with the highest or lowest values of the Y coordinate is given higher priority in an alternating fashion. This is to maximize the area used in filling the turbines.

Once the point is chosen, the wake region of the turbine is superimposed onto the velocity plot using the V-GS formula. Points in the neighbourhood of the new turbine are given the value 0 in the Territory matrix in accordance with the distance constraint. The greedy function keeps on choosing points for planting subsequent turbines and terminates when the Nth turbine is chosen.

For the given example below, S = 1000, N = 8, and D = 100. For the plots given, white regions represents 1 and black regions represents 0.

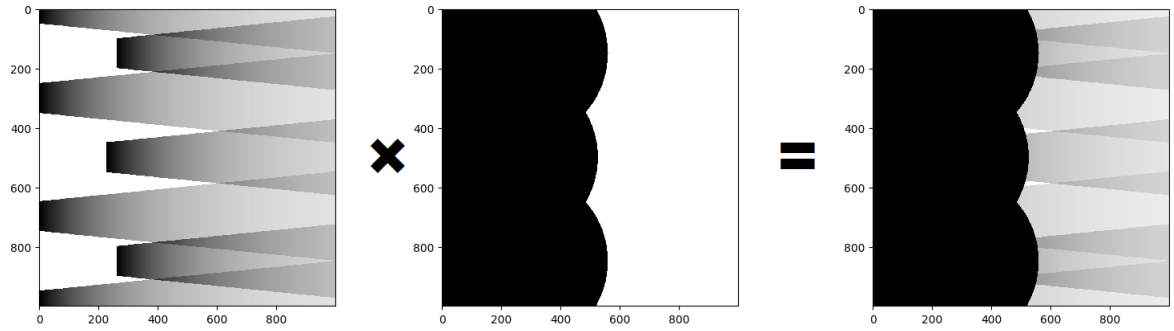

Figure 3: Element-wise multiplication of territory and velocity matrices

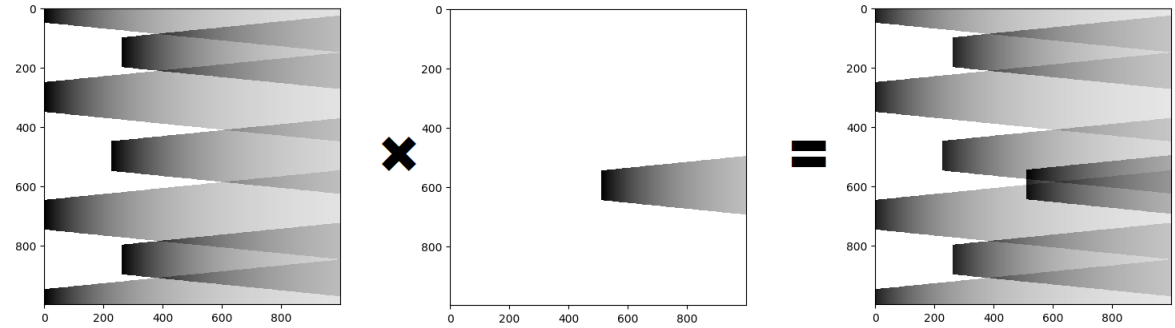

Figure 4: Updating the velocity matrix using the V-GS superposition principle

4.2 Pattern Search Algorithm

The Pattern Search Algorithm is a numerical optimisation technique used for solving unconstrained optimisation problems. The basic idea behind this algorithm is to iteratively search for the best solution by testing different design variables and evaluating their performance. The algorithm starts with an initial guess of the design variables, and then it explores the design space by making small perturbations to the variables. The perturbations are made in a specific pattern, hence the name "Pattern Search". The algorithm evaluates the performance of each perturbation and selects the one that improves the performance the most. This process is repeated until a satisfactory solution is found.

In the case of wind farm layout optimisation, the design variables are typically the location and orientation of wind turbines. The goal is to find a layout that maximizes the power generation of the wind farm while minimizing the wake effect caused by the turbines. The wake effect occurs when the wind passing through a turbine slows down, creating a "wake" behind it that can reduce the performance of downstream turbines.

The Pattern Search Algorithm [3] can help to minimize the wake effect by exploring different turbine layouts and orientations. However, it can be sensitive to the choice of search pattern or neighbourhood and may converge slowly or get stuck in local optima if the search pattern is not appropriate. In the pattern search algorithm, for this problem, the approach amounts to moving each turbine in x, and then y, until an improvement to the total performance is found. When an improvement is found, that candidate point is accepted as the new design point, and the algorithm continues through the coordinate directions, looking for improved points. If the algorithm proceeds through an entire cycle of 4N moves (N being the number of turbines) without

finding any improvement, then the step size is halved and the process continues. The algorithm is stopped when the step size is less than 0.001.
One advantage of the Pattern Search Algorithm is that it does not require any gradient information, which can be difficult to obtain for complex optimisation problems. Instead, it only requires the ability to evaluate the objective function (in this case, the power output of the wind farm) for each perturbation. This makes it a useful tool for wind farm layout optimisation, which is a complex and nonlinear problem.

5 Experiments

For the following experiments, we considered a grid of 3142 x 3142 consisting of 30 turbines i.e., $S = 3142$, $N = 30$ & $D = 100$.
Google Colab Links for the Python Codes: Link

5.1 Particle Swarm Optimisation

500 particles were initialised and r1, r2 weights were set randomly for them. Objective function was chosen to incorporate the output of the wind farm as well as a penalty term of -500 for every instance of 'incorrect' placement like going out of the grid or setting the turbines too close. The number of iterations for which the objective function converges was observed to be 100 hence that was used for all subsequent experiments. Initially the hyperparameters were set to a fixed value of w = 0.5, c1 = 0.9 and c2 = 0.5, but later inspired by [1] we set a decay so that exploration is encouraged in initial iterations but exploitation is encouraged later.
It was found that the choice of initial configuration affects the performance of the PSO algorithm drastically. Initially the particles were initialised randomly, but the objective function was negative due to a lot of mis-classifications. Hence, a pseudo random process of initialising the particles was implemented where the turbine configurations are chosen such that they are already spaced out from other turbines. Further, the output from the **greedy initial configuration** strategy was fed into the PSO for more optimisation. This approach improved the performance further and the final result is shown in the image.

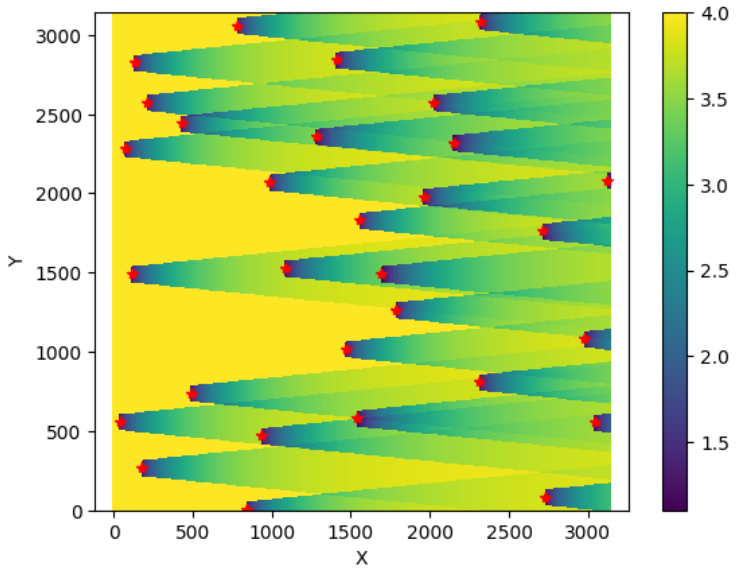

Figure 5: Resulting Wind Farm layout after Particle Swarm Optimisation

Experiments were also conducted for comparing configurations with a higher number of turbines,

around 75 turbines and with a lower number of turbines. It was found that for a lower number of turbines, greedy algorithms provided the optimum solution but for a higher number of turbines PSO provided further optimisation.

5.2 Pattern Search

For this part of the experiment, the wind farm was considered to be a grid of $\pi \times \pi$ having 30 turbines, with D = 0.1. This is a scaled down version of the parameters that we used for the PSO implementation.
In this algorithm, an initial layout of the wind farm was generated using a random function. Then at a time, the x or y coordinate of a turbine was moved by a value of 2.048 initially, if changing the coordinate produced an increase in the efficiency then that change was taken into account in the layout if it does not produce any increase in the efficiency then the other coordinate was changed and was checked for the increase in efficiency. If iterating over all the turbines and changing the coordinates (2×30) does not produce an increase in the efficiency then the step size with which we were changing the coordinates of the turbines was decreased by a factor of 2. Now, each turbine coordinate was moved by a new step size, and the wind farm layout was checked for any increase in efficiency, this process was repeated until the step size became 0.001, which was set as a step break. There were some functions in the code which were used by the main function "pattern_search" among which one was "jensen_wakes" which modelled the wake region of the turbines and calculated the effective velocity of the turbines according to the Jensen model which was discussed above, this function also plotted the wind farm layout and returned the total power of the current wind farm layout. There was a function "constraint" that had two hyperparameters rho1=10 and rho2=10 and returned a penalty term based upon the distance between the turbines and the end boundary grid constraint, there was a function "turbine_distances" which returned a 30×30 matrix containing Euclidian distance between any two pair of turbines. The "pattern_search" function took both total power and penalty term into account in deciding whether the current wind farm layout is better than the previous one. This approach improved the wind farm efficiency and gave us a layout having an efficiency of 95.104%.

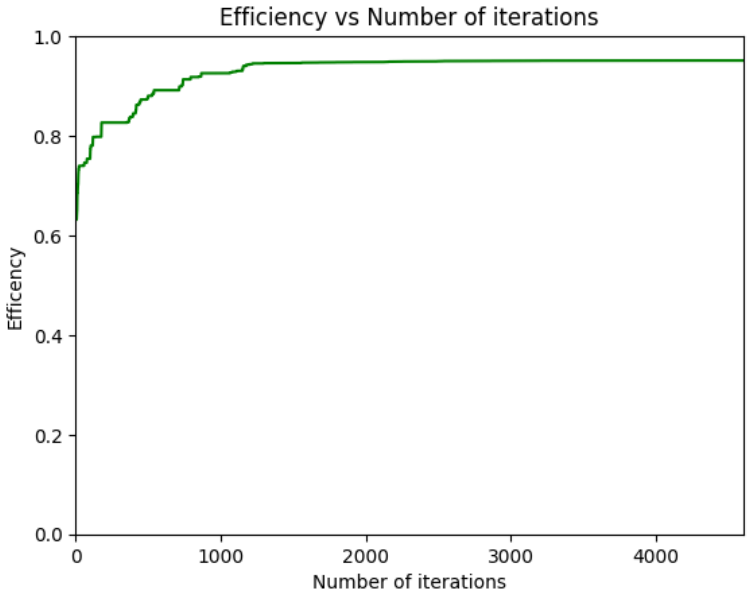

Figure 6: Efficiency vs no. of iterations for the algorithm

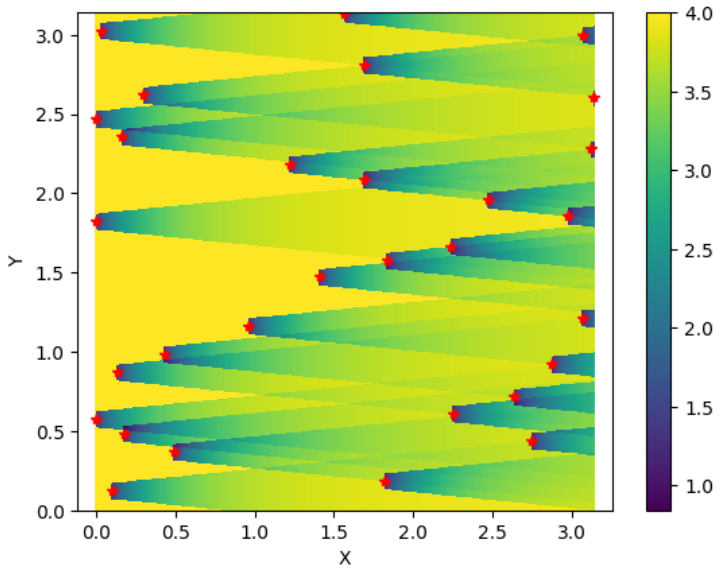

Figure 7: Resulting Wind Farm layout after implementing Pattern Search Algorithm

6 Conclusions

- **The Pattern Search algorithm achieved a better efficiency,** while PSO also exhibited a respectable performance.
- For PSO we found that the final solution depends heavily on the choice of initial configurations. Making the initial configuration randomized but subjected to the constraint on inter-turbine distance improved the efficiency.
- Overall, these algorithms were successful in solving the problem of finding the optimal windmill layout and it shows that this kind of treatment is pertinent since even a small increase in efficiency will greatly improve the overall production of wind farms.

7 Future Work

- Implementation of the Adaptive Simulated Annealing Algorithm: This can overcome some of the limitations of the Pattern Search Algorithm which were as follows - a turbine cannot be moved in x and y directions simultaneously, two turbines cannot be moved at once, turbines are allowed to move only in discrete step sizes.
- Taking into account the whole wind rose (following a Gaussian distribution) instead of having a uniform wind velocity.
- The current mathematical model is in 2D, whereas the wind farm industry typically employs 3D models, indicating an opportunity for enhancement.

8 Contributions

Rahul Patel (200100127) (20%): Pattern Search, Mathematical modelling of constraints, Modelling single turbine
Vinayak Gautam (200020161)(20%): Pattern Search, Mathematical modelling of constraints, Modelling single turbine

Amal Thomas (200100020) (20%): Greedy Algorithm, Modelling single turbine, Mathematical modelling of constraints
K S Varun (200100088) (20%): Greedy Algorithm, Modelling single turbine, Mathematical modelling of constraints
Aayush Shah (190100003) (20%): Particle Swarm Optimisation, Programming the model in Python

References

[1] M. Clerc and J. Kennedy. "The particle swarm - explosion, stability, and convergence in a multidimensional complex space". In: *IEEE Transactions on Evolutionary Computation* 6.1 (2002), pp. 58–73. DOI: 10.1109/4235.985692.

[2] N. O. Jensen. "A note on wind generator interaction, Ris-M, No. 2411". In: (1983).

[3] Jacob R West. "Optimizing Wind Farm Layout for Maximum Energy Production Using Direct Methods". In: (2020).