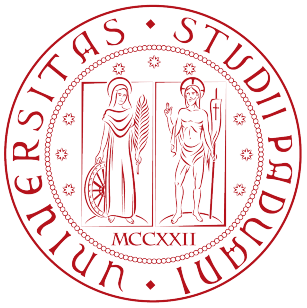


UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria dell'Informazione - DEI
Corso di Laurea Magistrale in Ingegneria dell'Automazione

Mixed Integer Programming Models and Algorithms for Wind Farm Layout

Laureanda: Martina Fischetti

Realatore:
Prof. Michele Monaci

Correlatori:
Prof. Jakob Stoustrup
Prof. John-Josef Leth
Dr. Anders Bech Borchersen

Anno accademico 2013-2014

Ai miei genitori

Acknowledgements

This thesis has been developed in collaboration with Aalborg University (Denmark) at the department of Electronic Systems, Automation & Control. I had the pleasure to meet my Danish supervisors already during my Erasmus program in Aalborg, and I would like to thank them to propose me to come back for this interesting thesis. During my thesis work I also had the opportunity to get in touch with a great company as Vattenfall, where I could learn and get in touch with lots of people and new ideas. I would like to thank all the people who helped me for the duration of this thesis, and throughout my university career.

I really want to thank all my supervisors who followed me in this thesis adventure, both in Italy and in Denmark. I learnt a lot from this international collaboration and from my thesis in general. In particular I would like to thank:
Prof. Michele Monaci for following me from Italy and for always being proactive in suggesting new ideas and methods for the optimization problem.
Prof. Jakob Stoustrup who proposed the thesis topic and introduced me to this fascinating problem. Thanks for believing in me from the start and for always being positive in the thesis development.
Prof. John-Josef Leth for following my work with passion and interest. When any problem occurred, you were always direct and very clear in all your suggestions.
Dr. Anders Bech Borchersen for following me and supporting me in the cooperation with Vattenfall.

From the Vattenfall company, I would like to thank Benjamin Martinez who helped me with the CFD simulation data and with WindPRO, and Jens Madsen who has always been interested in the development of my work.

Last but not least, I would really like to thank my family and all my friends, both in Italy and in Denmark, who supported my growth, each in their own way.

Contents

Abstract

The interest in green energy has recently grown all over the world as the demand for energy is increasing and, at the same time, old energy resources, fossil fuels in particular, are becoming rare, expensive, and pollutant. Many countries such as Germany, United Kingdom, Sweden, and Denmark are investing in renewable energy.
In particular Denmark (where this thesis has been developed) is one of the most active countries in green energy research. In February 2011, indeed, the Danish Parliament has approved a really ambitious plan: it states that by 2020, 35% of the energy produced shall be renewable energy, and 100% by 2050.

Since the wind energy field of research is relatively new and of great interest, there is a big need of decision support tools and new ideas to solve the design problems that are emerging every day. In this challenging context the present thesis is focusing on the optimal design of wind farm layout. This problem has been studied in collaboration with Vattenfall, a Swedish company leader in the wind energy sector in all the North of Europe.
The original results of the thesis have been considered of interest both from a company point of view (as our tool turned out to outperform the commercial software that is currently used at Vattenfall), and from a research point of view, as we presented a new approach able to deal with instances with huge amount of variables (10000+ possible turbine positions) within a few minutes on a standard PC.

1 Introduction

The goal of the thesis is the development of a decision support tool for the following problem:

Given a specific site, its resource maps and terrain constraints,
determine a feasible allocation of turbines that maximizes power production, (1.1)
subject to turbine proximity and turbine wake constraints.

The decision support tool to be developed must be able to solve the problem in both onshore and offshore scenarios.

Figure 1.1. Example of wind farm onshore (left) and offshore (right)

To better understand statement (1.1) we briefly sketch the onshore example that we are going to study in greater detail in Chapters 2 and 6.

We received in this case the characteristics of a site located in United Kingdom, the wind distribution in the site, and the characteristics of the turbines that should be located (Siemens SWT-2.3-93). Many constraints should be taken into account to define the actually feasible positions – for example, terrain inclination, presence of buildings, turbulence, extreme wind speed, etc. These requirements resulted in a constraint map built by Computational Fluid Dynamics (CFD) simulations. The final possible positions to install a turbine were given on input to our optimization tool. After a few minutes of computing, our tool determinates the best feasible coordinates to locate turbines in the site. The overall process is sketched in Figure 1.2.

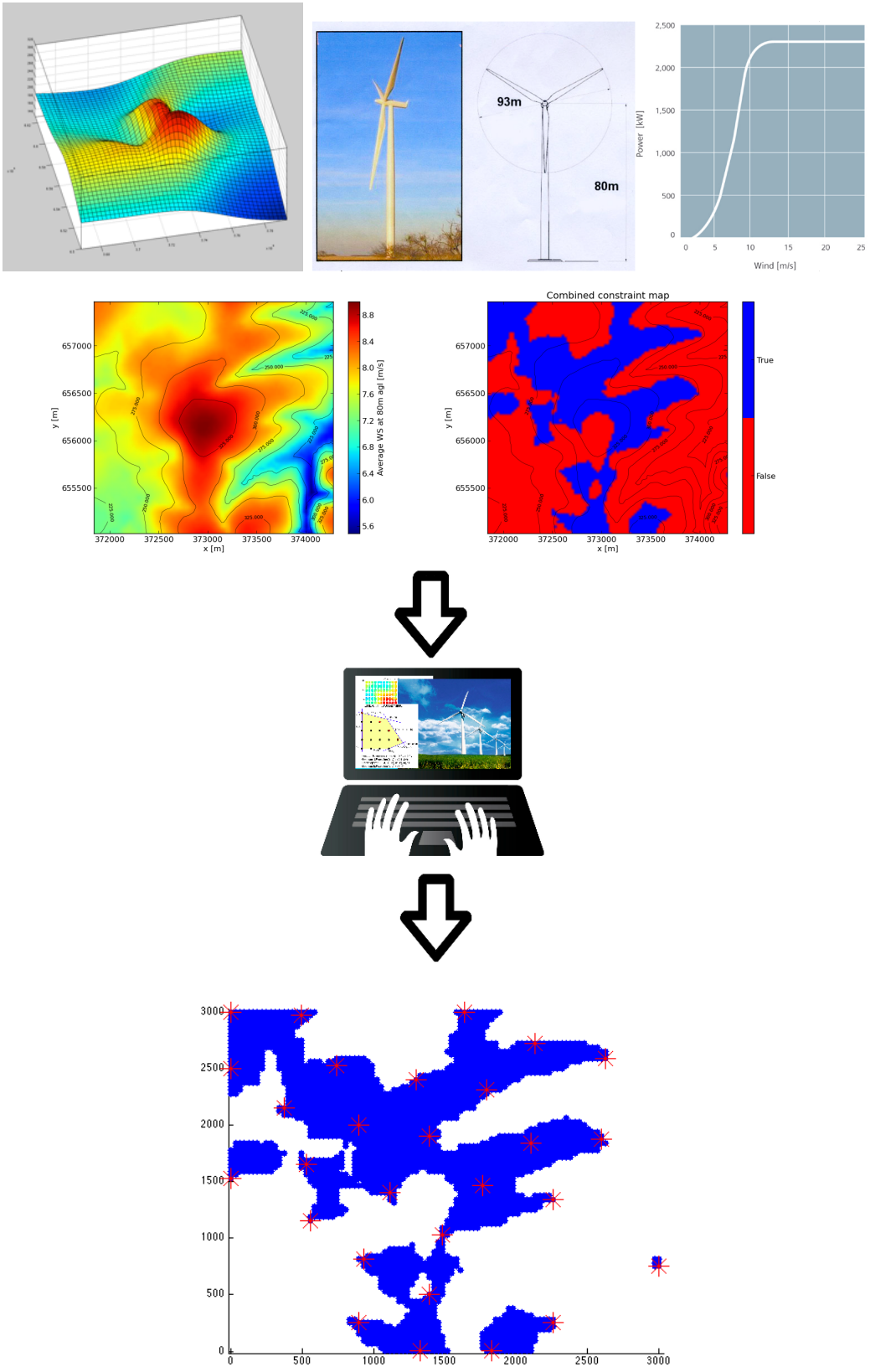


Figure 1.2. Illustration of a decision support tool for optimal turbine allocation. **Input** (from left to right, top to bottom): 1) characteristics of the site; 2) Siemens SWT-2.3-93 turbines characteristics (with power curve); 3) wind distribution in the site (the figure shows the Average Wind Speed for the site, as an example); 4) feasible positions (blue area of the constraint map) in the site determined by additional constraints on the site. **Output** (bottom plot): Optimal positioning of turbines within the site.

A key aspect of our tool is that it takes wake effect into account when determining the optimal layout. The wake effect is the interference phenomenon for which, if two turbines are located one close to another, the upwind one creates a shadow on the one behind. This is of great importance in the design of the layout since it results into a loss of power production for the turbines downstream, that are also subject to a possibly strong turbulence. The figure below, that refers to the Horns Rev 1's offshore wind farm (see [1] for more details), well illustrates the problem.

Figure 1.3. Wake effect in the Horns Rev 1 wind farm [2]

As it can be seen, under this wind scenario only the first turbine line is hit by full wind, while all the other turbines are suffering for strong wake effect. A layout like the one in the picture should then be penalized in our search of the optimal layout.
As a result, we needed to model interference between turbines. We used Jensen's model [3], that we adapted for the onshore case. Our model of the interference takes several possible wind scenarios (obtained from real-world samples) into account, weighed by their probability.

A what-if analysis has been carried out to study how the optimal layout changes when considering different wind scenarios and different maximum number of turbines.

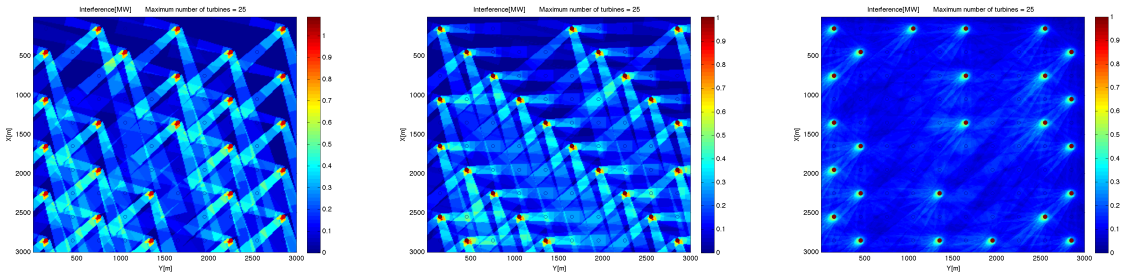


Figure 1.4. Optimal layout solutions considering a maximum number of turbines equal to 25, under 4, 10 and 500 wind scenarios. Red circles represent built turbines, while color in the background refers to the interference induced by those turbines

We modeled the layout optimization problem by using the Mixed Integer Linear Programming (MILP) paradigm, and solved the resulting model by means of ad-hoc heuristic algorithms.
The MILP approach has been applied to wind farm problems only recently. Models from the literature [4; 5] have been analyzed and new MILP models and heuristics have been developed. The overall approach that we propose in this thesis is a combination of initial ad-hoc heuristics

(1-opt and 2-opt) and of MILP heuristics (Proximity Search) on suitable MILP models. This new approach turns out to be particularly competitive for large-scale instances, outperforming the standard MILP approach (that uses the MILP solver alone), the heuristics from the commercial MILP solver IBM ILOG Cplex 12.5.1 [6], and the 1-opt and 2-opt algorithms alone. With the use of this new method our tool is able to deal with 10000+ possible positions within a few minutes on a standard PC.

Finally, the layouts designed by our tool have been compared with the output of the commercial optimization software (*WindPRO*) currently used at Vattenfall. Tests shows that our tool is able to outperform the competitor both in the case of fixed and of unlimited maximum number of turbines.

The thesis is organized as follows.

Chapter 2 gives a precise definition of the problem we consider, both in the onshore and offshore settings. The test cases are introduced, describing their constraints in details.

Chapter 3 addresses how to model interference. The well-known Jensen's model is first described, and then extended to deal with 3D cases.

Chapter 4 describes the main Mixed Integer Linear Programming models developed for wind farm layout optimization in the literature, along with their Stochastic Programming counterparts dealing with uncertainty in wind speed and direction. A new model is also introduced, that overcomes the issues arising with previous proposals– mainly, the possibility of dealing with the very large scale instances arising in practical applications.

Chapter 5 introduces the solution methods we have developed and implemented. Ad-hoc 1-opt and 2-opt heuristics based on a fast parametric computation are first introduced. More sophisticated heuristics based on the MILP model are then presented, and combined to obtain a powerful hybrid heuristic.

Chapter 6 reports our tests. A first illustrative example is first reported, showing how the optimal turbine placement changes depending on different wind scenarios and maximum number of turbines. A real-world onshore test case is then addressed. Extensive computational results for very large scale offshore random cases are also reported, with a comparison among alternative solution techniques.

Chapter 7 compares our tool with a commercial software widely used by practitioners, showing the benefits deriving from the use of our techniques.

Chapter 8 draws some conclusion and addresses future work.

Problem definition 2

2.1 General problem

The layout of wind farms is a topic of interest for many wind energy companies. It is, indeed, a challenging problem because it implies difficult constraints, where a better layout translates into a huge profit (in terms of energy and money).

The entire layout design is a complex process that includes many steps, many of which are still carried out manually. The whole process, indeed, includes a large number of constraints of different types: physical constraints, law constraints, visual/noise constraints, and so on. What is done nowadays is essentially to compute an optimal solution from a wind resources point of view, and then to change it manually to satisfy all the other constraints. This of course requires a lot of expertise and a lot of time, and cannot lead to a proven optimal solution. Due to the enormous amount of information that must be handled, there is of course plenty of room for computer aided optimization.

This thesis focuses on the optimal layout from a wind-resource point of view; possible extensions, like optimizing the cable layout, will be mentioned in Chapter 8.

The scope of our tool, generally speaking, is finding a layout that maximizes the power production by considering the wind distribution in the site and terrain constraints. As of today, commercial software for the solution of this specific problem is available. However, its performance is not completely satisfactory from an optimization point of view. One of these commercial optimization tools will be presented later in this thesis.

Figure 2.1. Proximity constraint. The red circles represent a rotor diameter: typically the minimum distance between turbines is around four or five rotor diameters.

A first restriction to be considered is the proximity constraint (Figure 2.1), i.e. two turbines cannot be placed too close one to each other otherwise they will physically clash. Typically the minimum distance between turbines is around four or five rotor diameters. The reason to ensure a sufficient spacing between turbines (larger than just 1 or 2 rotor diameters) is to reduce their interactions, i.e., heavy wind turbulence between turbines [7].

2.2 Offshore scenario

Term *offshore* refers to the construction of wind farms in bodies of water, so the site is normally a regular area in a sea zone. This has some advantages, for example the wind is generally stronger and equally distributed. In addition, residents opposition to construction is usually much weaker, so the so-called NIMBY ("Not In My Back Yard") problem is easy to solve in this case. However, some other costs should be taken into account, such as cable and foundation costs depending on the sea depth.

In the offshore case, the possible positions to be considered are generally on a regular grid. The depth of the water is supposed to be equal everywhere, so the offshore case is actually a 2D case. Figure 2.2 shows an example of a grid of 100 possible positions in a square area of 3000m x 3000m

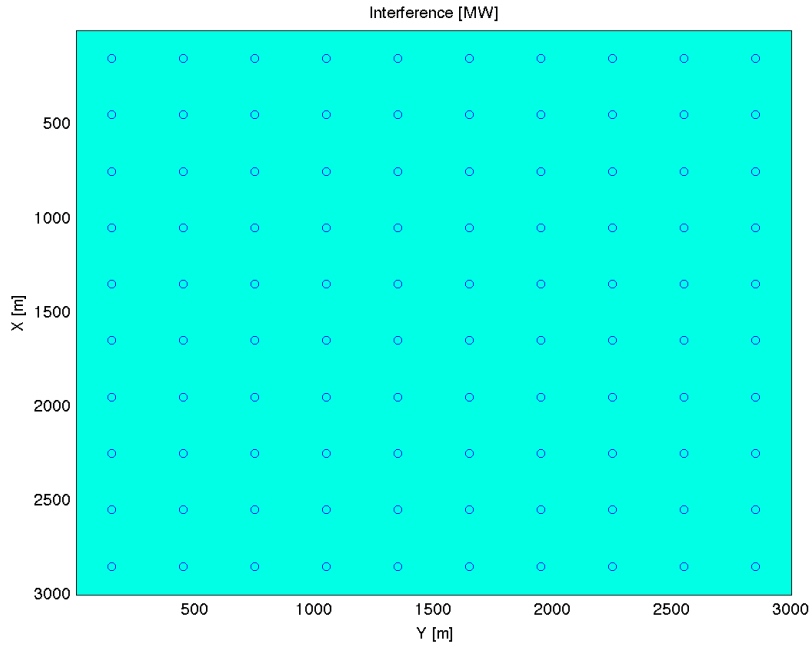


Figure 2.2. An example of an offshore area of 3000m x 3000m. Blue circles represent the possible positions to locate turbines.

Of course a better grid resolution, i.e. more possible positions to locate turbines, can result into a better wind farm design. However, this implies to cope with a larger number of variables, and more computational resources are needed.

Another input for the offshore case is the wind distribution in the site, i.e., a time series of wind samples with their speeds and directions. Note that, in an offshore setting, wind can be considered to be equally distributed all over the site, so the potential power production of all possible positions is the same. A key importance has then the interference between turbines, that causes a loss of production in some positions. Figure 2.3 shows the interference distribution over the site when two turbines are built on the grid: in the dark blue area (where there is no wake effect) all possible positions are equivalently good. In the positions where the interference is greater than zero, instead, a turbine would produce less power. The optimization tool will

therefore place the turbines to avoid the zones with interference.

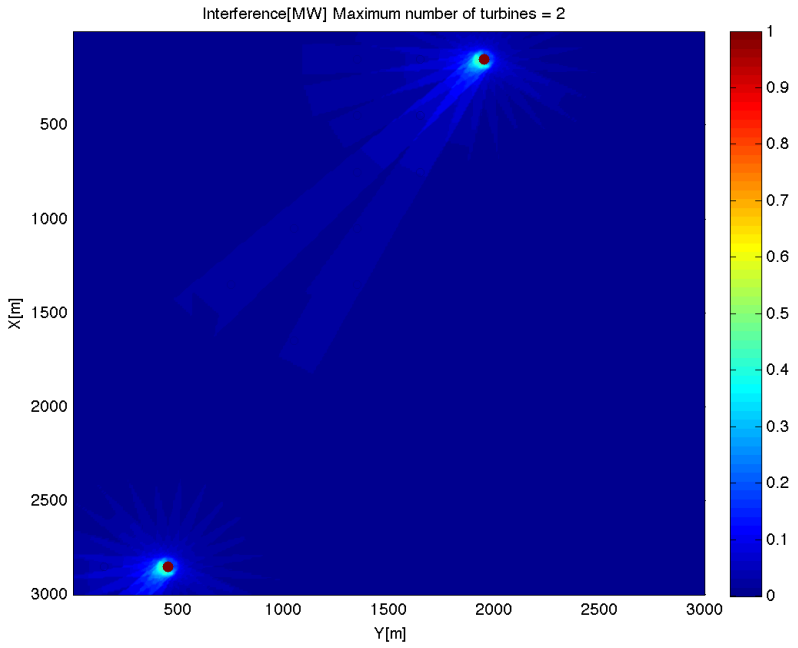


Figure 2.3. Optimal layout solutions imposing maximum number of turbines equal to 2. Red circles represent built turbines, while the color in the background refers to the interference (power loss) induced by those turbines.

2.3 Onshore scenario

Term *onshore* refers to the construction of wind farms on lands, where many more constraints should be taken into account. To introduce them we refer to our test case: a 2000m x 2000m real-world site located in United Kingdom.
First of all, the site can include some mountains and hills, so also the z-coordinates should be taken into account and the problem becomes a 3D problem.

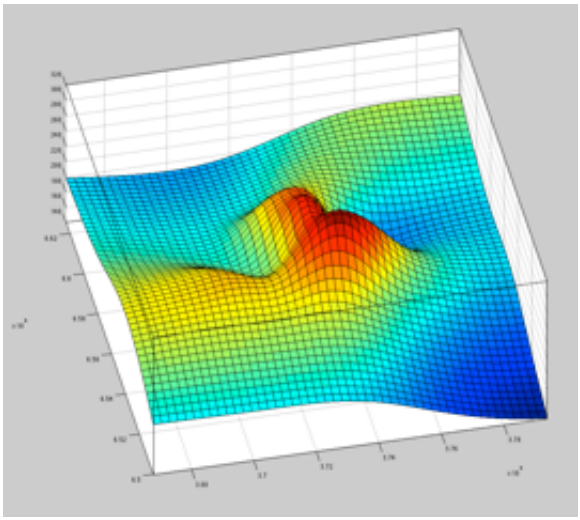


Figure 2.4. A 3D plot of a real-world landscape that we will use as test case in Chapter 7

Due to the different heights and to the presence of obstacles, even without any interference wind

is no longer uniform in the site. Figure 2.5 plots average wind speed obtained by Computational Fluid Dynamics (CFD)[1] simulations. It is easy to see from this plot how wind speed changes significantly from point to point.

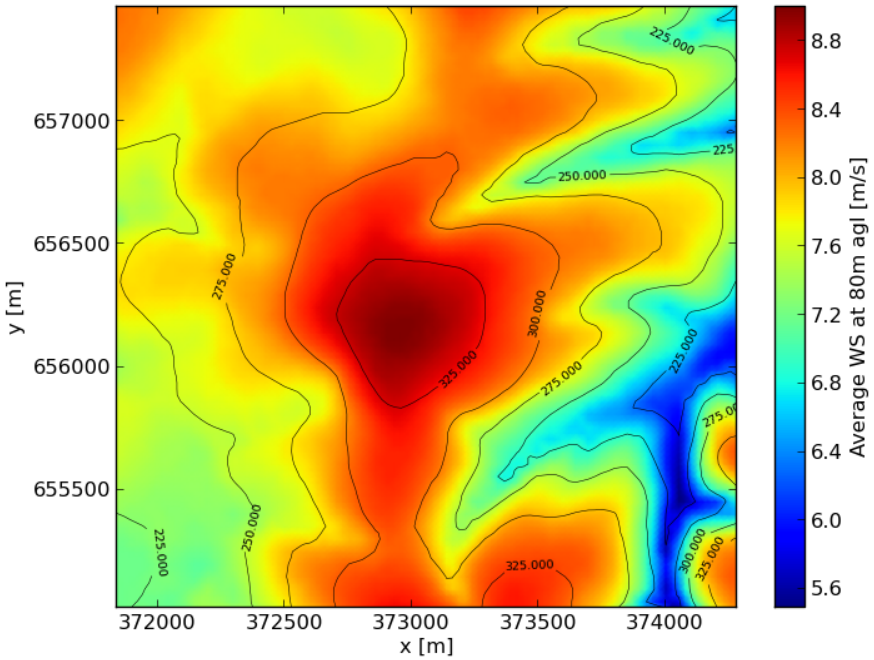


Figure 2.5. Average wind speed [m/s] at 80m agl (above ground level) for our test site

Wind direction is influenced by the presence of mountains. The following figures show maximum and minimum wind flow inclination.

[1]CFD is a branch of fluid mechanics that uses numerical methods and algorithms to study problems that involve fluid flows, such as wind related problems

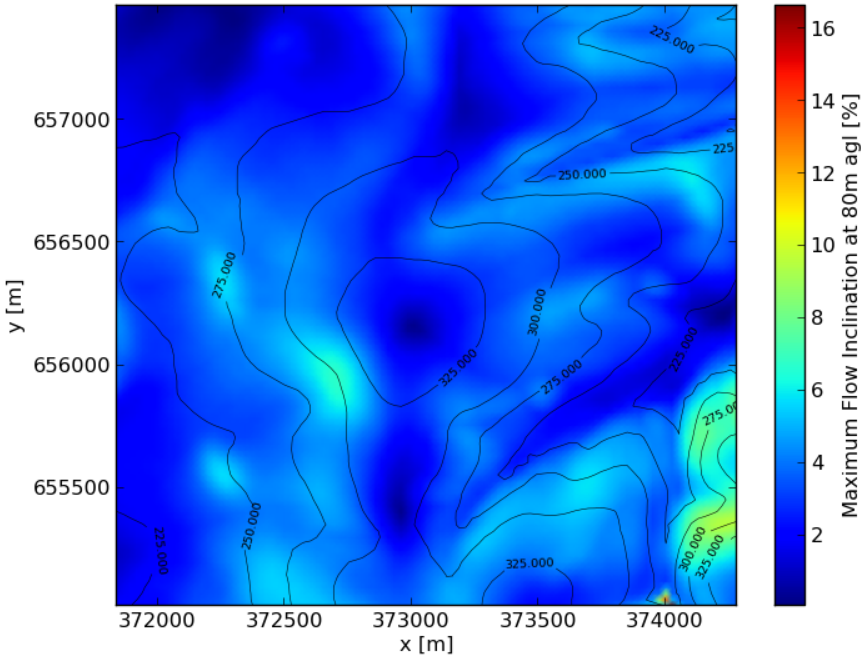


Figure 2.6. Maximum Flow Inclination [m/s] at 80m agl for our test site

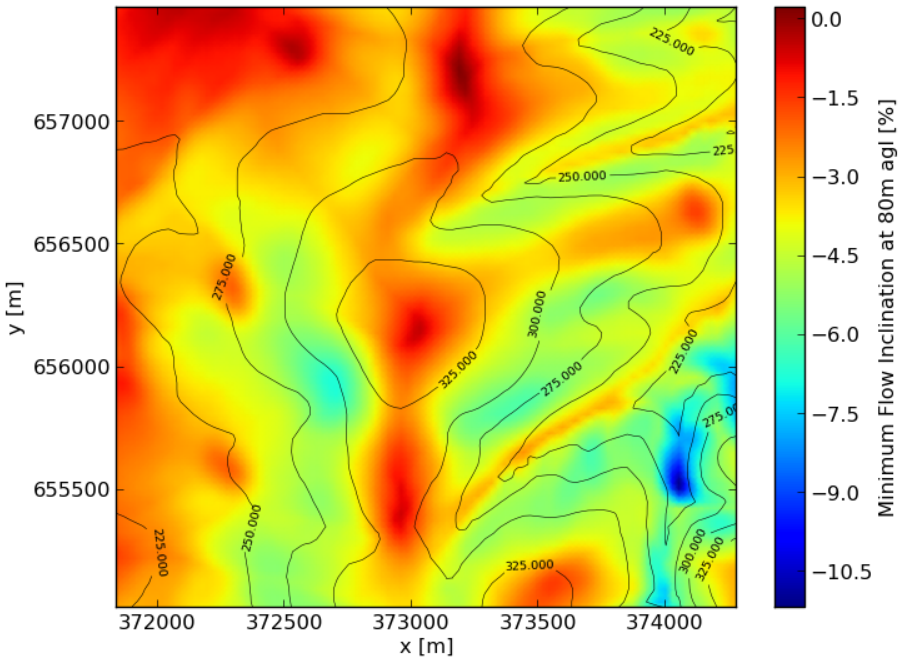


Figure 2.7. Minimum flow inclination [m/s] at 80m agl for our test site

All the above result in a different power distribution within the site (even without considering interference) and, consequently, in different interference values depending on the actual turbine positions.

Moreover, in the onshore case not all the positions in the site are actually feasible due to terrain constraints. In our test case, for example, the following constraints (shown also in Figure 2.8) are considered:

- **Extreme wind**: if the wind is too strong the turbines risk to be damaged. Points where the average wind is too strong must be avoided.
- **Turbulence**: wind turbulence increases the loads on the blades, gearbox, generator and tower. In addition, turbulent wind can excite resonances (large vibrations) in the turbine and in its supporting tower. All those loads can cause fatigue problems in turbine structures.
- **Flow inclination**: when wind turbines are placed on steep slopes or cliffs the wind might hit the rotor not perpendicularly. A large in-flow angle could lead to a high level of fatigue for the turbine.
- **Shear coefficient**: the shear coefficient represents the presence of obstructions in the area, such as buildings, trees, etc. If the shear coefficient is too large a position cannot be used .
- **Wind distribution compliance**: Zones where the wind is too strong could not be feasible positions for the selected turbines, according to the IEC 61400 standards. IEC 61400 is a set of wind turbines standards required to ensure the engineering integrity of wind turbines, providing an appropriate level of protection against damage from all hazards during the planned lifetime [8]. According to IEC, turbines are divided in classes depending on how much wind speed they can well sustain.
- **Terrain inclination**: excessive terrain inclination could be problematic for the installation of the turbine and can cause high fatigue on the turbine structure.

All together, those constraints give the feasible positions for turbines, represented in blue in the combined constraint map (last plot in Figure 2.8).

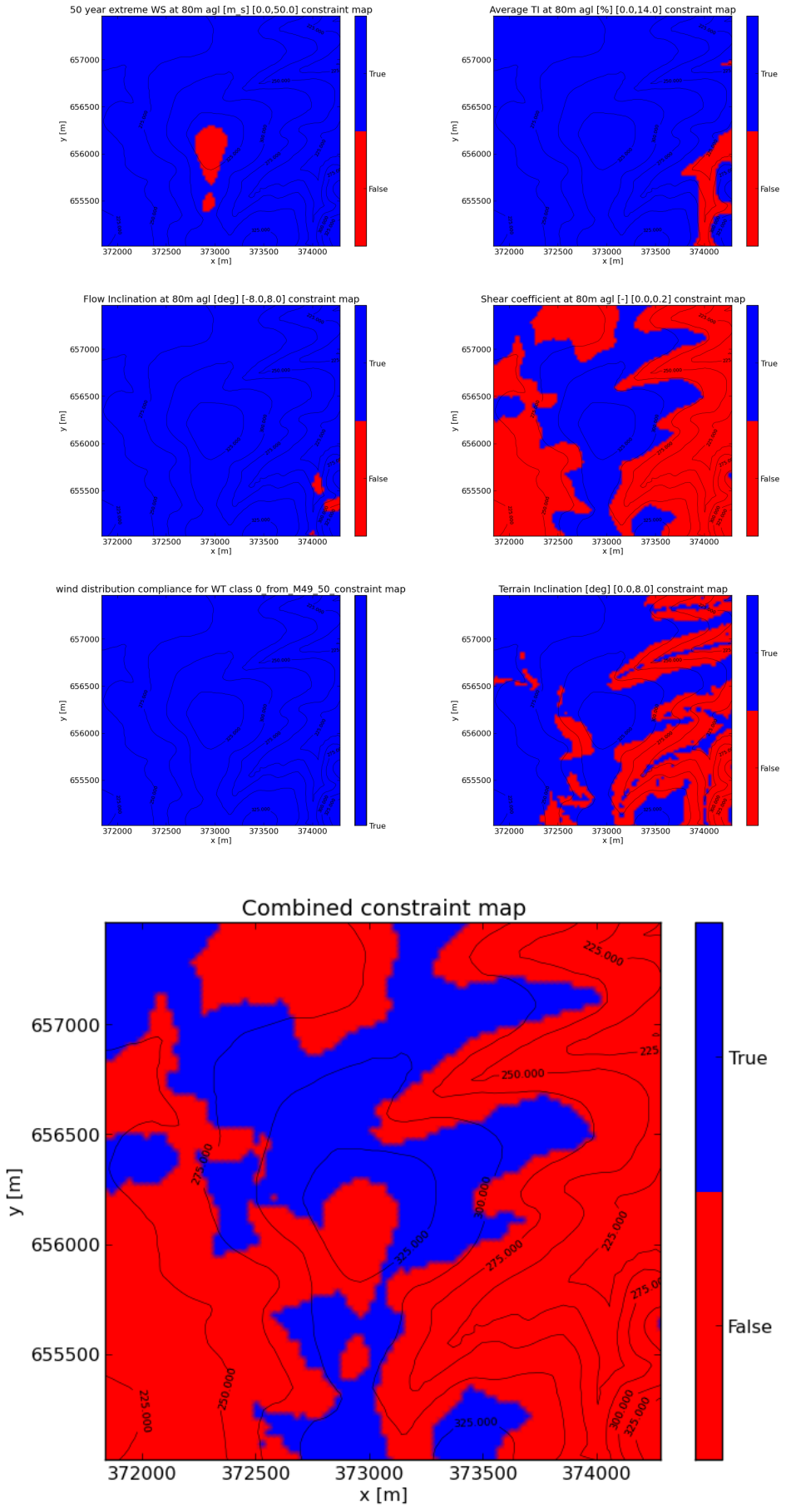


Figure 2.8. Constraint maps obtained by CFD simulations for our real-world site: respectively, extreme wind speed, turbulence intensity, flow inclination, shear coefficient, wind compliance for the WT class in exam, and terrain inclination. All together, those constraint maps give the possible positions for wind turbines, represented in blue in the last plot.

Modeling interference 3

As already mentioned, interference between turbines has to be taken into account in our optimization models. The wake effect, indeed, can cause a strong reduction in the final power production. Note however that too complicated (and computationally heavy) models for the interference are impractical for large cases.
In this chapter a simple way to quantify interference is described.

3.1 Jensen's model

The model proposed by N.O. Jensen [3] is a simple way to describe the wake effect between two turbines. This model is also used in WindPRO[9], an industry-standard software for wind resource assessment and placement of wind turbines within wind farms.

Let us consider two generic turbines, say i and j, such that i is the upwind one and j is the downstream one with respect to a specific wind direction. Let U denote the speed of the wind that affects i. Jensen's model describes the wake effect behind i as a trapezium that has, as its minor base, the rotor of the turbine i of dimension, say, D (see Figure 3.1)

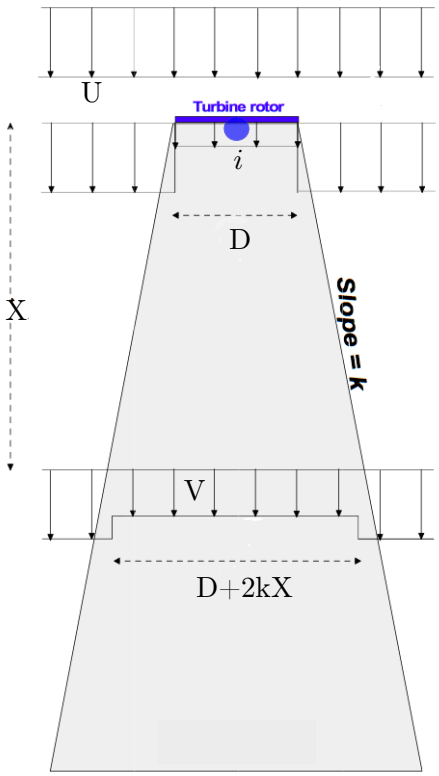


Figure 3.1. Wake effect according to Jensen's model with wind from north to south

Each point inside the trapezium (gray zone in the figure) is affected by the wake due to the presence of i. The loss of wind speed for a turbine (in the trapezium) at distance X from i, can

be computed as

$$\delta V = U(1 - \sqrt{1 - C_t})(\frac{D}{D + 2kX})^2 \tag{3.1}$$

Where: U = upwind speed at turbine i
C_t = thrust coefficient corresponding to wind speed U
D = rotor dimension (diameter)
X = distance between i and j (on the x-axis)
k = wake decay constant, typically

$$k = \begin{cases} 0.075 & \text{onshore} \\ 0.050 & \text{offshore} \end{cases}$$

The wind speed at j is computed as $V = U - \delta V$. The interference $I_{i,j}$, defined as the loss of power on j due to i, can then be computed by using the turbine power curve (Figure 3.3) according to the following table:

Wind Speed (m/s)	Power (MW)	C_t
3.00	0.000	0.00
4.00	0.065	0.81
5.00	0.180	0.84
6.00	0.352	0.83
7.00	0.590	0.85
8.00	0.906	0.86
9.00	1.308	0.87
10.00	1.767	0.79
11.00	2.085	0.67
12.00	2.234	0.45
13.00	2.283	0.34
14.00	2.296	0.26
15.00	2.299	0.21
16.00	2.300	0.17
17.00	2.300	0.14
18.00	2.300	0.12
19.00	2.300	0.10
20.00	2.300	0.09
21.00	2.300	0.07
22.00	2.300	0.07
23.00	2.300	0.06
24.00	2.300	0.05
25.00	2.300	0.05

Figure 3.2. Power and thrust coefficient C_t as a function of wind speed

Note that both thrust coefficient C_t and power depend on the upwind speed in a non-linear way.

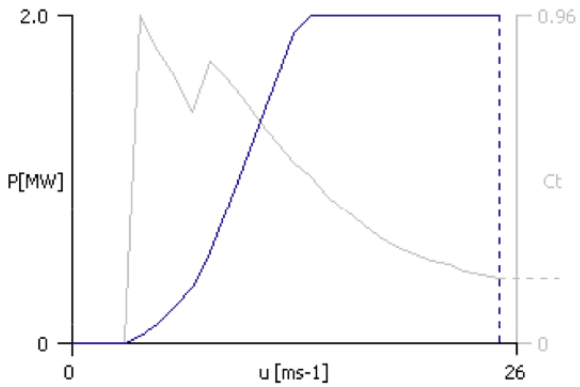


Figure 3.3. Wind Speed vs C_t (gray line) and Wind Speed vs Power (blu line) [10]

The interference that a turbine causes using the Jensen's model has been plotted below. The upwind is supposed to come from north. Note that, when the upwind is 15.0 m/s, there is an effect of saturation in the power curve hence the difference of power (i.e. the interference) is really small (it goes from 0 to 0.01, see Figure 3.4). Instead, when the upwind is 7.0 m/s the interference is stronger (it goes from 0 to 0.5, see Figure 3.5)

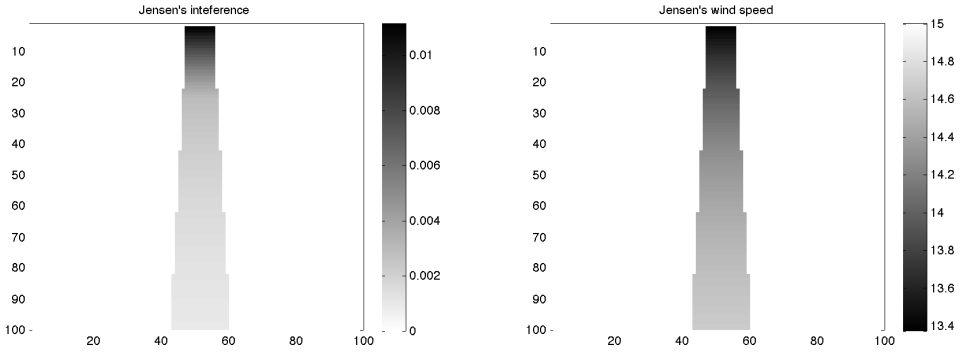


Figure 3.4. Interference and wind speed computed using Jensen's model; pixel = 10x10m; wind = 15.0 m/s from north

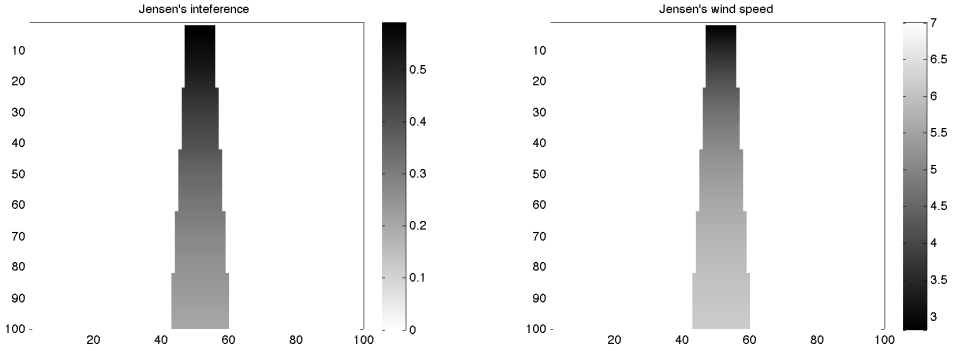


Figure 3.5. Interference and wind speed computed using Jensen's model; pixel = 10x10m; wind = 7.0 m/s from north

3.2 Jensen's model for a generic wind direction

This section shows how to update Jensen's formula for a generic wind direction. In particular we need some easy calculations to know whether a given point j is in the wake produced by an upwind turbine i, when wind direction is not necessarily vertical. Consider then Figure 3.6, where wind is not parallel to the x-axis. Note that the rotor automatically moves to become perpendicular to the wind direction.

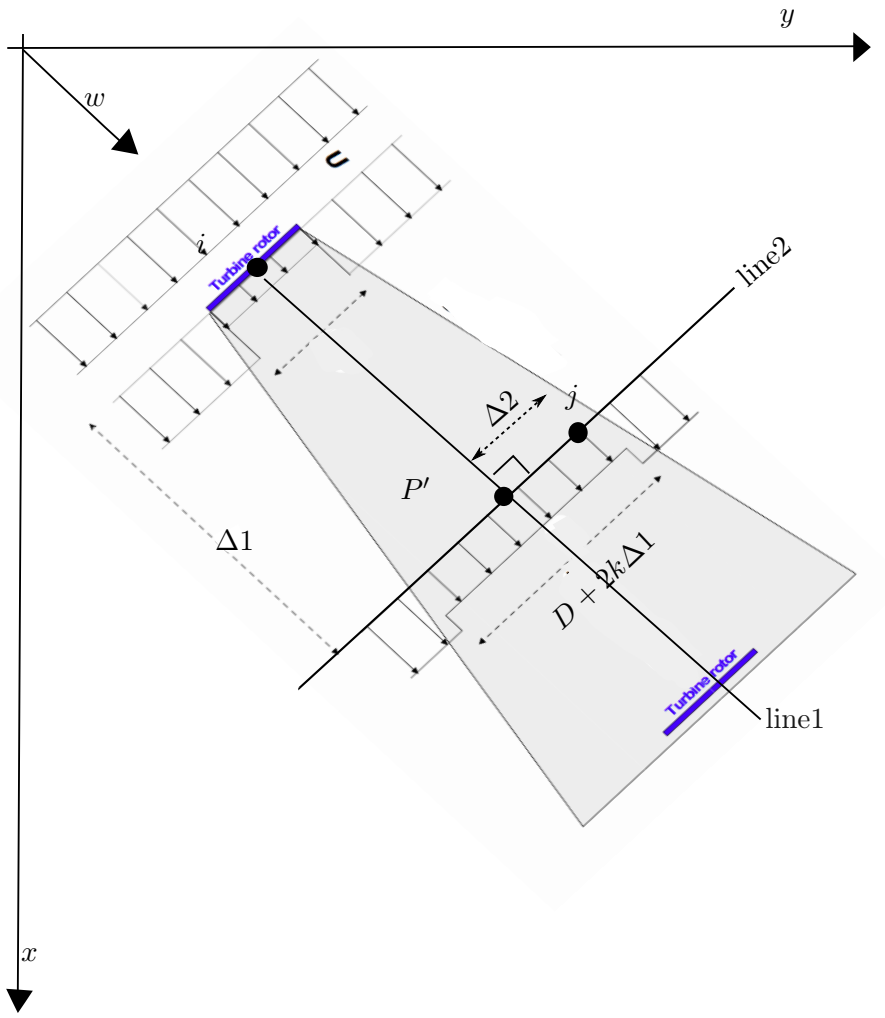


Figure 3.6. Wake effect modeled as in Jensen's model for wind coming from north-west

In the figure, w is a vector that describes wind direction, while $i, j \in \mathbb{R}^2$ are the two points where turbines are located. In order to decide whether j is in the wake of i, we need to determine the position of point P' in the figure, and to compute $\Delta 1 = \text{dist}(i, P')$ and $\Delta 2 = \text{dist}(j, P')$. Then j is in the shade trapezium iff $\Delta 2 \leq \frac{D+2k\Delta 1}{2}$.

In Figure 3.6, the line that passes through i and goes in the wind direction is called line1 and is parametrized by α (say), while the line that passes through j and is perpendicular to line1 is called line2. P' is the intersection point between line1 and line2 so it satisfies both the equation that describes line1 (equation 3.2) and the equation for the perpendicularity of the two lines (equation 3.3):

$$P' = i + \alpha w \tag{3.2}$$

$$P' - i \perp j - P' \qquad \text{i.e. } (P' - i)^T (j - P') = 0 \tag{3.3}$$

Solving this system of equations leads to:

$$\alpha = \frac{(j - i)^T w}{||w||^2} \tag{3.4}$$

Given α, point P' can easily be computed by using equation (3.2).

Note now that if $\alpha \leq 0$ then i is upstream, so the interference on j due to i is zero (i.e. $I_{i,j}$=0). If $\alpha > 0$ instead we have to test if j is in the trapezium, i.e., if $\Delta 2 \leq \frac{D+2k\Delta 1}{2}$. In this case, interference can be computed according to formula (3.1), with $\Delta 1$ instead of X.

The plots in Figure 3.7 as in Section 3.1, with a wind direction $w = (0.5, 0.5)$ and speed 8 m/sec.

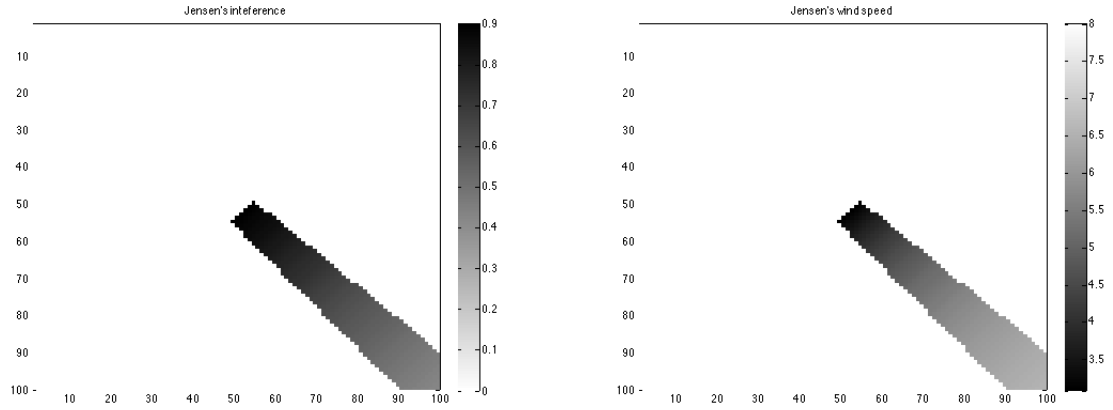


Figure 3.7. Interference and wind speed computed using Jensen's model; pixel = 10x10m; wind = 8.0 from north-west

3.3 Weighted interference for the MILP models

Interference (computed as explained in the previous sections) will be used in the objective function of our mathematical models. Wind is however not stable and changes its direction and intensity over time. Therefore in our models we will use the average interference weighed by wind probability.

Plots in Figure 3.8 show the average interference in case of three possible wind situations:
1. wind (1.00, 0.00), speed 7 m/s, probability 0.30
2. wind (0.50, 0.50), speed 8 m/s, probability 0.60
3. wind (0.00, 1.00), speed 9 m/s, probability 0.10

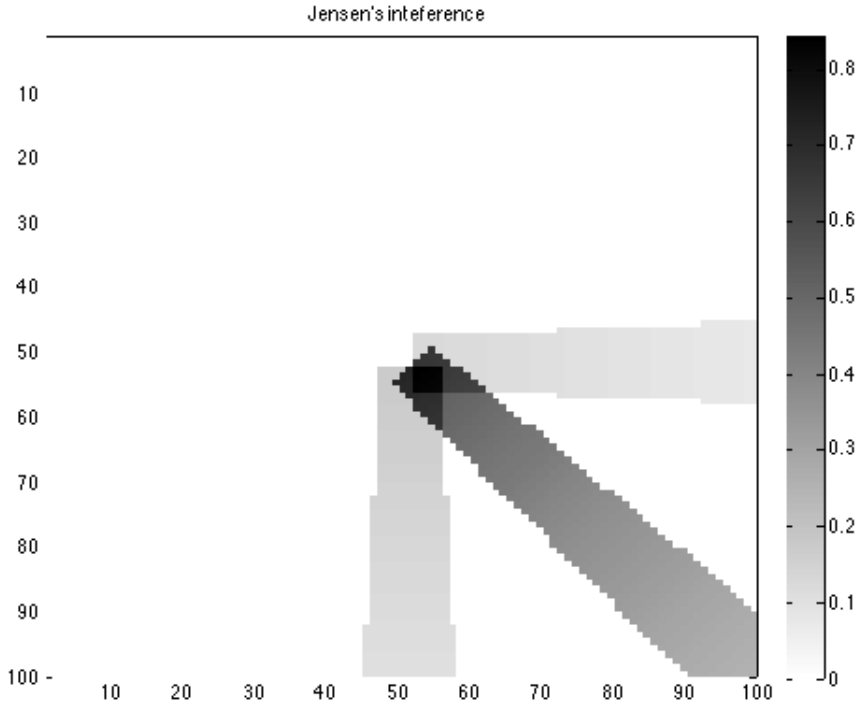


Figure 3.8. An example of $I_{i,j}$ computed as the average interference between three possible wind scenarios: wind = 7.0 m/s from north with probability 0.3, wind = 8.0 m/s from north-west with probability 0.6, wind = 9.0 m/s from west with probability 0.1 ; pixel = 10x10m;

Figure 3.9 shows the average interference calculated on real-world frequency series from the European Wind Energy Association EWEA (250,000+ real-world wind measurements). They have been divided in bins (24 sectors of 15 deg each for direction and in 1m/s bins for wind speed), obtaining about 500 macro-scenarios.

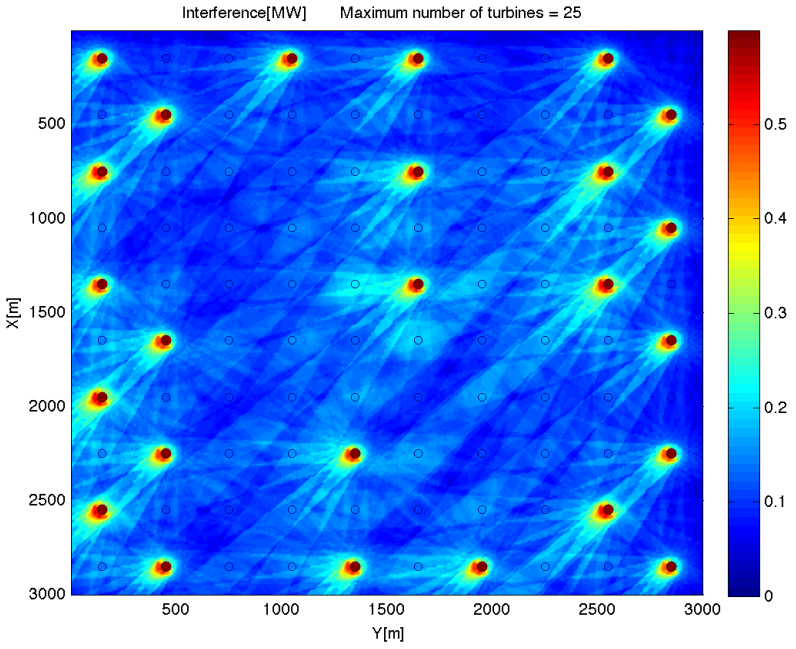


Figure 3.9. Average interference with all different wind scenarios from the EWEA data

3.4 Interference in a 3D case

Jensen's model working with general wind direction can also be used in a 3D scenario. That means that, under proper assumptions, a Jensen's model can be considered in an onshore setting as well. Nevertheless, determining interference for onshore problems is actually much more difficult than in the offshore (2D) case, due to the presence of nonuniform wind.
Due to possible different heights, indeed, even without any interference, wind is no longer uniform in the site (using the notation of Section 3.2, U is now different from point to point). We refer to Figures 2.5, 2.6 and 2.7 in Chapter 2 for some illustrative plots in a real-world case.

Some assumptions need be formulated to be able to extend Jensen's model to a 3D scenario. First at all, we are not considering wind flow inclination. That means that the interference cone is always horizontal and does not depend on the shape of mountains and hills. Due to this assumption, our 3D wake cone would look like the one in Figure 3.10.

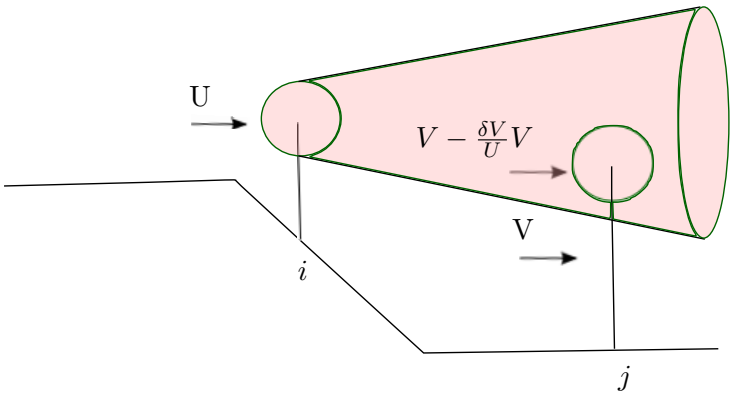


Figure 3.10. A simple scheme for our 3D Jensen's model

The wind speeds U and V are the average wind speed given on input for points i and j respectively (i.e., we take the values shown in Figure 2.5 for the real-world test site). Because of this, value δV computed by (3.1) cannot be used as it is, as this could even produce a negative wind value at point j. Therefore a proportion has been used to compute wind reduction at point j due to the presence of a turbine at point i. To be more specific, δV is still calculated according to (3.1) but the wind reduction applied in j is $\frac{\delta V}{U} V$, i.e., the new wind in j is now computed as

$$V' = V - \frac{\delta V}{U} V \tag{3.5}$$

Wind direction is also different from point to point. However, considering a wind direction time series for each point would be computationally too heavy, so we consider only one of such series to compute the average interference (as explained in Section 3.3).

Mixed Integer Linear Programming Models

4

We next briefly sketch some main concepts and definitions about Linear and Mixed Integer Linear Programming; the reader is referred to, e.g., the book of Nemhauser and Wolsey [11] for a deeper treatment of the subject.

4.1 LP and MILP: the main idea

A convex problem is a mathematical programming problem

$$\text{minimize } f(x) : x \in X \tag{4.1}$$

where f and X are convex. Those problems are characterized by the fact that each local optimum is also a global optimum. Linear Programming (LP) is a subclass of convex problems in which both the constraints and the optimization criterion are linear functions. That means, in particular, that we can write an LP problem in a matrix form as follow:

$$\begin{aligned} \text{minimize} \quad & c^T x && (4.2)\\ \text{subject to} \quad & Ax = b && (4.3)\\ & x \geq 0 && (4.4) \end{aligned}$$

where $x = (x_1, x_2, \cdots, x_n)$ are the variables of the problem, c is a vector of coefficients, A is a $m \times n$ matrix, and b is the right-hand-side vector. (4.2) is called the objective function, while (4.3) express the feasibility constraints. An LP problem has an interesting geometrical representation: its constraints define a set of hyperplanes and halfspaces in $\mathbb{R}^n$, so satisfying all of them leads to a polyhedron [1]. We will indicate the polyhedron of all the feasible solutions as P. It can be proven that, assuming that P is bounded, at least one vertex of the polyhedron P is optimal.
A main technique to solve LP problems is the Simplex method, invented by George Dantzig in 1947. This method begins with a starting vertex of P, and moves along the edges of the polyhedron by possibly improving the objective function, until it reaches an optimal vertex.

[1] A polyhedron is in fact defined as the intersection of a finite number of hyperplanes and/or halfspaces.

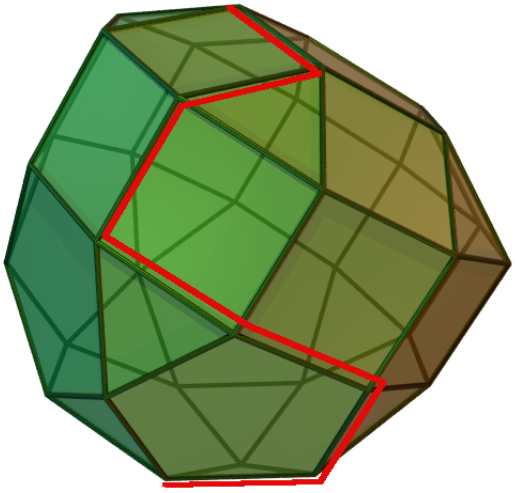


Figure 4.1. The simplex algorithm moves along the edges of the ployhedron until it reaches an optimal.

An Integer Linear Programming problem (ILP) is an LP problem where all the variables are required to be integer. A Mixed Integer Linear Programming (MILP) problem is an LP problem where some of the variables are required to be integer (some others are possibly allowed to be noninteger). The main techniques developed for this problems are explained in the next section. As we will see, the problem of determining an optimal wind farm layout can be formulated as a MILP problem and then solved with the use of a black-box MILP solver.

4.2 MILP

As introduced in Section 4.1, an ILP is a problem that can be written as

$$\begin{aligned} \text{minimize} \quad & c^T x && (4.5)\\ \text{subject to} \quad & Ax = b && (4.6)\\ & x \geq 0 && (4.7)\\ & x \ \ \textit{integer} && (4.8) \end{aligned}$$

The integrality requirement (4.8) makes the problem non-convex. The main idea used to solve ILP problems is then to consider the problem without the integrality condition, i.e., to solve the so-called *continuous relaxation* of the problem. Let us define the pair (A, b) in (4.6) as a *formulation*, and let P denote the associated polyhedron. Note that the formulation is not unique. In particular two formulations $(A, b) \rightarrow P$ and $(\tilde{A}, \tilde{b}) \rightarrow \tilde{P}$ are equivalent if $P \cap \mathbb{Z}^n = \tilde{P} \cap \mathbb{Z}^n$. Among all possible equivalent formulations, a one producing an inclusion-minimal P is called ideal. It can be proved that, if we choose the ideal formulation, then the continuous relaxation polyhedron P^* (say) is in fact the convex hull of all the points in $P \cap \mathbb{Z}^n$. Then, an optimal vertex of P^* found when solving the relaxed problem is always integer, thus it is an optimal solution solution for the ILP as well.

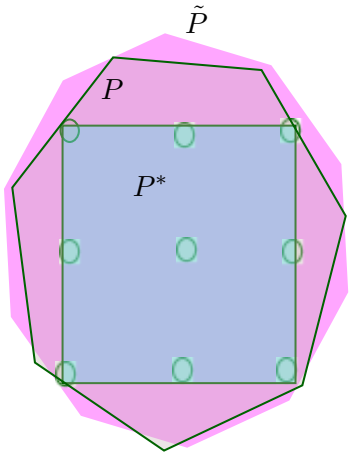


Figure 4.2. Three equivalent formulations. Dots represent $P \cap \mathbb{Z}^n = \tilde{P} \cap \mathbb{Z}^n = P^* \cap \mathbb{Z}^n$. P^* (blue polyhedron) is the convex hull

Unfortunately, in most cases the ideal formulation is huge and very time consuming to obtain. Therefore, we would like to get to an approximation of the convex hull P^*. The way to do that is to improve step by step P until we get a sufficient approximation of P^*. In terms of constraints, this means to append some new constraints to the initial formulation that make some of the noninteger solutions in $P \setminus P^*$ infeasible. This operation is called "generation of cuts" because the new constraints act as cuts in a geometric interpretation (see Figures 4.3).

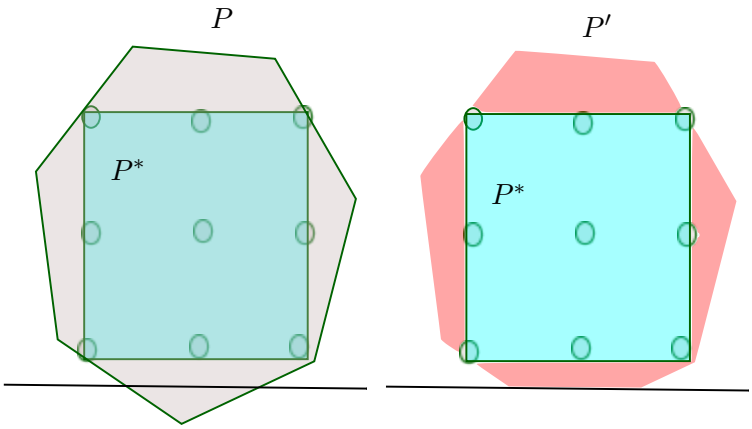


Figure 4.3. A cut is generated to get P closer to P^*. The new polyhedron P' (in red in the second figure) obtained from P using this cut is closer to the convex hull P^*

A solution algorithm based on systematic generation of cuts is called Cutting Plane method. The practically effectiveness of this approach, however, would require the capability of generating "strong" cuts, which is problematic in practice. Consider, for example, polyhedron in Figure 4.4. P can be reduced to the convex hull P^* in few steps by the use of few strong cut (figure on the right) or in a large number of steps when using less effective cuts (figure on the left).

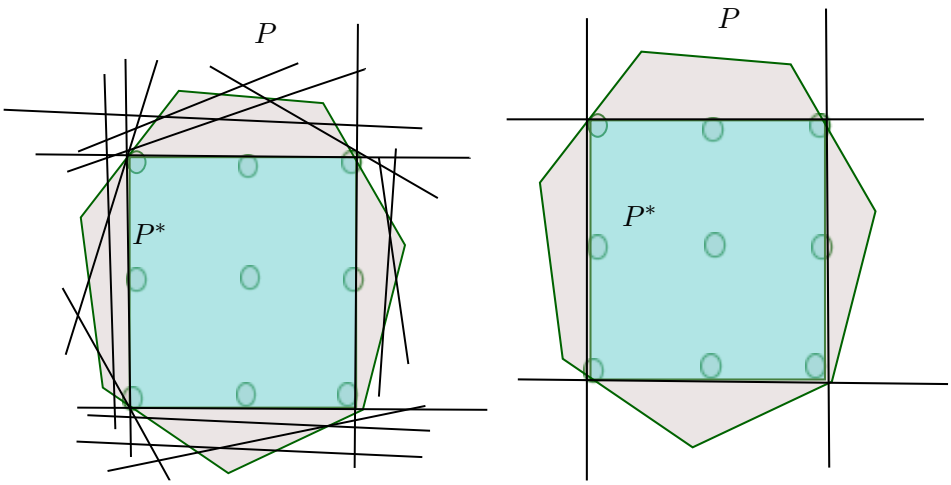


Figure 4.4. The importance of generating strong cuts to get convex hull P^*

An alternative approach to solve an ILP problem is called *Branch & Bound*. The idea is to develop a search tree that, at each node, solves the continuous relaxation of the ILP; see Figure 4.5 for an illustration. More specifically, we are given the original ILP, say ILP_0 $\min\{c^T x \ : \ Ax = b, \ x \geq 0, \ \ x \ \ integer\}$. We can relax it to LP $\min\{c^T x \ : \ Ax = b, \ x \geq 0\}$ and solve it, getting an optimal vertex x^*. If x^* is not integer, we select a noninteger component of x^*, say x^*_h, and solve recursively the two ILPs obtained by introducing the branching condition $x_h \leq \lfloor x^*_h \rfloor$ and $x_h \geq \lceil x^*_h \rceil$, respectively.

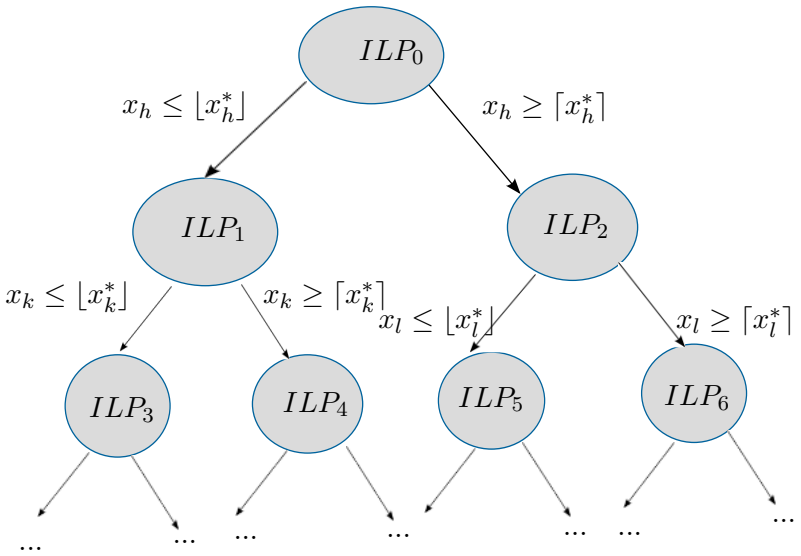


Figure 4.5. Branching tree

Assuming that P is bounded, the number of nodes in the Branch& Bound tree is finite, but potentially extremely large—for a problem with n binary variables about 2^n nodes can be generated in the worst case.

One of the key points to make the Branch & Bound method faster is to prune the subtrees that will not get to an optimal integer solution, i.e. "fathom" them as soon as possible. The algorithm to do that works as follows: the incumbent solution value z_{opt} (i.e. the cost of the best integer solution so far), is stored. At each node, an optimistic estimate of the optimal solution value (lower bound at the node) is computed as the optimal value of the LP relaxation, and

compared with z_{opt}. If the lower bound is greater than or equal to z_{opt}, then the node cannot lead to a solution better than the incumbent one, so it can be fathomed. See Figure 4.6 for an illustration.

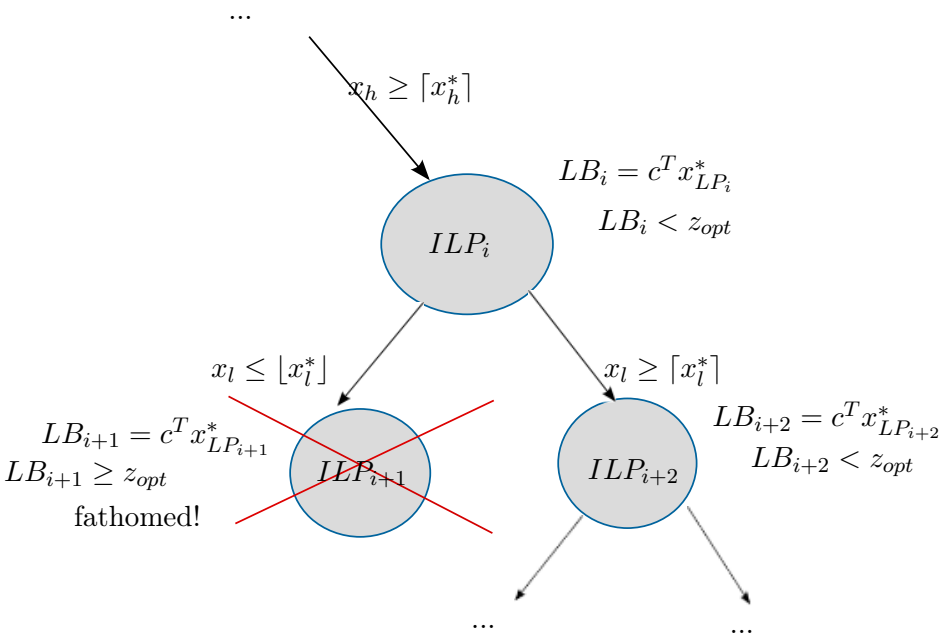


Figure 4.6. Branching tree with fathoming

Of course, the most frequent the fathomings the faster the Branch & Bound method. For that reason, advanced solvers apply several heuristics to produce good values of z_{opt} early in the tree.

The *Branch & Cut* algorithm is an effective way to combine Cutting Planes and Branch & Bound. It consists in the development of a Branch & Bound tree, but at some nodes a certain computing time is spent to generate cuts with the aim of improving the lower bound. How much time is actually spent at each node depends on the quality of the generated cuts: as soon as they do not improve the lower bound in a significant way, the method resorts to a classical branching step.

4.3 Literature: MILP approaches in wind farm layout optimization

Our optimal wind farm layout problem can be modeled as a MILP problem. The main idea is to describe the layout problem as a vertex packing problem using binary/integer variables. Let us then consider all possible positions where a turbine can be built as the vertices of a graph $G = (V, E)$. Edges in E represent relationships between turbine, i.e., turbine proximity and interference.
In the following three models from literature (mainly from [4; 5]) are studied.

4.3.1 First model: how to model turbine proximity

The first basic model that has been developed wants to maximize the power production considering just two basics constraints:

- the maximum number of turbines that can be built is equal to a given value N_{MAX}

- there should be a minimal separation distance between two turbines to ensure that the blades do not physically clash (turbine proximity constraint)

To write this simple ILP model a binary variable x_i is defined for each possible position $i \in V$ such that:

$$x_i = \begin{cases} 1 & \text{if a turbine is built at position } i \\ 0 & \text{otherwise} \end{cases} \qquad (i \in V)$$

The model would then be:

$$\text{maximize} \sum_{i \in V} P_i x_i \tag{4.9a}$$

subject to

$$\sum_{i \in V} x_i \leq N_{MAX} \tag{4.9b}$$

$$x_i + x_j \leq 1 \qquad \forall i, j \in V : \; j > i, \;\; \text{dist}(i,j) \; < D_{MIN} \tag{4.9c}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{4.9d}$$

Where: P_i is the power that a turbine would produce if built (alone) in position i
N_{MAX} is the maximum number of turbines that can be built
D_{MIN} is the minimal separation distance

The objective function (4.9a) maximizes power production, without considering any loss due to interference. The first constraint (4.9b) limits the maximum number of turbines that can be built to N_{MAX}. Lastly, (4.9c) impose that, if the positions i and j are too close, not both the turbines can be built (x_i and x_j cannot be both 1).

Remembering the graph interpretation of the problem, let $G_I = (V, E_I)$ denote the incompatibility graph with

$$E_I = \{[i,j] : i, j \in V, \, dist(i,j) < D_{MIN}, \, j > i\}$$

and let $n = |V|$ denote the total number of sites. It could be noticed that the proximity constraints (4.9c) could be strengthened to their clique[2] counterpart

$$\textstyle\sum_{h \in Q} x_h \leq 1 \qquad \forall Q \in \mathcal{Q} \tag{4.10}$$

where $\mathcal{Q}$ is a family of maximal cliques [3] of G_I, such that every edge in E_I is covered by at least one member of $\mathcal{Q}$.

[2] A clique in a graph $G = (V, E)$ is a set $Q \subseteq V$ such that for every two vertices in Q, there exists an edge connecting the two.

[3] A maximal clique is a clique that cannot be extended by including one more vertex.

We decided, in this model and in the followings, not to improve the proximity constraints to their clique form (4.10), for two main reasons: (i) the number of cliques can be exceedingly large, and (ii) usually, a family of cliques is automatically generated during preprocessing by any MILP solver, which can control their number and can handle them very efficiently. As we have no additional information about which cliques should be generated explicitly, we preferred to leave clique management to the solver itself.

4.3.2 Second model: considering interference

The second model is an expansion of the first model that considers also the wake effect between two turbines, hence modifying the objective function (4.9) to $\sum_{i \in V} P_i x_i - \sum_{i \in V} \sum_{j \in V} I_{i,j} x_i x_j$. In other words, if two turbines i and j are both built, the interference between those two will cause a loss in the power production. The way to convert the quadratic objective function into a linear model is to define another set of binary variables $z_{i,j}$ such that

$$z_{i,j} = \begin{cases} 1 & \text{if turbines are built in both positions } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \qquad (i, j \in V,\, j > i)$$

The model would then be:

$$\text{maximize} \sum_{i \in V} P_i x_i - \sum_{i \in V} \sum_{j \in V : j > i} (I_{i,j} + I_{j,i}) z_{i,j} \tag{4.11a}$$

subject to

$$\sum_{i \in V} x_i \leq N_{MAX} \tag{4.11b}$$

$$x_i + x_j \leq 1 \qquad \forall i, j \in V :\ j > i,\ \text{dist}(i,j)\ < D_{MIN} \tag{4.11c}$$

$$x_i + x_j - 1 \leq z_{i,j} \qquad \forall i, j \in V, j > i \tag{4.11d}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{4.11e}$$

$$z_{i,j} \in \{0, 1\} \qquad \forall i, j \in V,\, j > i \tag{4.11f}$$

Where: $I_{i,j}$ is the interference caused by i to j
P_i is the power that a turbine would produce if built (alone) in position i
N_{MAX} is the maximum number of turbines that can be built
D_{MIN} is the minimal separation distance

The objective function (4.11a) now maximizes the power production considering the loss due to interference. Note the rule played by $z_{i,j}$: the interference $I_{i,j}$ will decrease the production only if both turbines i and j are built. The new constraint (4.11d) define variables $z_{i,j}$ in a linear way. Note, in fact, that the objective function would try to minimize $z_{i,j}$, so (4.11d) will set $z_{i,j}$ to 1 only if $x_i + x_j = 1$ (i.e., when turbines are built in both positions i and j). As a consequence, the two linearization constraints $z_{i,j} \leq x_i$ and $z_{i,j} \leq x_j$ need not be stated explicitly into the model.

It should be noticed that this model is overestimating the interference between all turbines. To see that, suppose to build three turbines, say i, j and k as in Figure 4.7

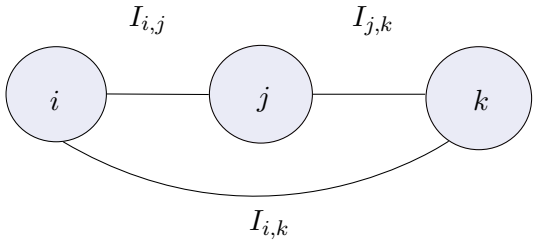


Figure 4.7. Interference between three turbines i, j and k

The interference from i to k is already counted in the loss of power given by $I_{i,j} + I_{j,k}$, however using model (4.11a) we are subtracting also $I_{i,k}$ from the total produced power. To avoid this overestimation a third model is proposed in the literature, where the only interference considered is the highest one.

4.3.3 Third model: interference not overestimated

The third model is similar to the previous one but only the largest interference w_i (say) caused by i to all its neighbors is considered in the objective function. That means that an explicit double-index variable $z_{i,j}$ is no longer necessary and can be substituted by

$$w_i = \begin{cases} \max\{I_{i,j} : \ j \in V, \ x_j = 1\} & \text{if } x_i = 1 \\ 0 & \text{if } x_i = 0 \end{cases} \qquad (i \in V)$$

The new model would then be:

$$\text{maximize} \sum_{i \in V} (P_i x_i - w_i) \tag{4.12a}$$

subject to

$$\sum_{i \in V} x_i \leq N_{MAX} \tag{4.12b}$$

$$x_i + x_j \leq 1 \qquad \forall i, j \in V : \ j > i, \ \text{dist}(i,j) < D_{MIN} \tag{4.12c}$$

$$I_{i,j}(x_i + x_j - 1) \leq w_i \qquad \forall i, j \in V \tag{4.12d}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{4.12e}$$

$$w_i \geq 0 \qquad \forall i \in V \tag{4.12f}$$

Note that constraint (4.12d) is a linear way to express the operation of taking the maximum interference possibly caused by position i.
As an illustration let us consider again the example in Figure 4.7. In that case equation (4.12d) would be:

$$I_{k,j}(x_k + x_j - 1) \leq w_k \tag{4.13a}$$

$$I_{k,i}(x_k + x_i - 1) \leq w_k \tag{4.13b}$$

Our hypothesis is that turbines i, j, k are all built, that mathematically corresponds to $x_i = 1$, $x_j = 1$ and $x_k = 1$. Then (4.13) becomes

$$I_{k,j} \leq w_k \tag{4.14a}$$
$$I_{k,i} \leq w_k \tag{4.14b}$$

In other words, w_k has to be not smaller than both $I_{k,j}$ and $I_{k,i}$, that implies that w_k will be set by the optimizer to the maximum between these two values.

4.4 Stochastic Programming variants

The definition of the turbine power vector (P_i) and of interference matrix (I_{ij}) depends on the wind scenario considered, that greatly varies in time. Using statistical data, one can in fact collect a large number K of wind scenarios k, each associated with a pair (P^k, I^k) and with a probability $\pi_k \geq 0$, with $\sum_k \pi_k = 1$. Using that data, one can write a Stochastic Programming [12] variant of the previous models where the objective is to maximize the average power production.

4.4.1 First model

The first model can easily be modified as follows:

$$\text{maximize} \sum_{k=1}^{K} \pi_k \left(\sum_{i \in V} P_i^k x_i\right) \tag{4.15a}$$
subject to
$$\sum_{i \in V} x_i \leq N_{MAX} \tag{4.15b}$$
$$x_i + x_j \leq 1 \qquad \forall i, j \in V : \; j > i, \;\; \text{dist}(i,j) \; < D_{MIN} \tag{4.15c}$$
$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{4.15d}$$

By defining the average power

$$\overline{P}_i := \sum_{k=1}^{K} \pi_k P_i^k \qquad \forall i \in V$$

the new objective function (4.15a) reads

$$\text{maximize} \sum_{i \in V} \overline{P}_i x_i \tag{4.16a}$$

so we are back to a model that has the same form as the original one, with uncertainty taken into account in the average P_i's. In other words, the stochastic programming variant of the first model just requires replacing powers with average powers in the objective function.

4.4.2 Second model

Analogously, the stochastic programming variant of the second model, can be written as follows:

$$\text{maximize} \sum_{k=1}^{K} \pi_k \left(\sum_{i \in V} P_i^k x_i - \sum_{i \in V} \sum_{j \in V : j > i} (I_{i,j}^k + I_{j,i}^k)\, z_{i,j} \right) \tag{4.17a}$$

subject to

$$\sum_{i \in V} x_i \leq N_{MAX} \tag{4.17b}$$

$$x_i + x_j \leq 1 \qquad \forall i, j \in V : j > i,\ \text{dist}(i,j) < D_{MIN} \tag{4.17c}$$

$$x_i + x_j - 1 \leq z_{i,j} \qquad \forall i, j \in V, j > i \tag{4.17d}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{4.17e}$$

$$z_{i,j} \in \{0, 1\} \qquad \forall i, j \in V \tag{4.17f}$$

As before, by defining the average power and interference as:

$$\overline{P}_i := \sum_{k=1}^{K} \pi_k P_i^k \qquad \forall i \in V$$

$$\overline{I}_{ij} := \sum_{k=1}^{K} \pi_k I_{ij}^k \qquad \forall i, j \in V$$

we can rewrite objective function (4.17a) as

$$\text{maximize} \sum_{i \in V} \overline{P}_i x_i - \sum_{i \in V} \sum_{j \in V : j > i} (\overline{I}_{i,j} + \overline{I}_{j,i}) z_{i,j} \tag{4.18a}$$

Also in this case, the stochastic programming variant is just the same as the original model, but considers average power and interference.

4.4.3 Third model

In this case, unfortunately, the stochastic programming variant becomes much larger than the original model, as we need extra variables and constraints for each scenario. The reason for this complication is that the third model needs to compute the maximum among interferences, and the maximum operation is nonlinear (the average of the maximum is not the maximum of the averages).

The stochastic programming version of the third model is

$$\text{maximize} \sum_{k=1}^{K} \pi_k \left(\sum_{i \in V} (P_i^k x_i - w_i^k) \right) \tag{4.19a}$$

subject to

$$\sum_{i \in V} x_i \leq N_{MAX} \tag{4.19b}$$

$$x_i + x_j \leq 1 \qquad \forall i, j \in V : \; j > i, \; \text{dist}(i,j) < D_{MIN} \tag{4.19c}$$

$$I_{i,j}^k (x_i + x_j - 1) \leq w_i^k \qquad \forall i, j \in V, \; \forall k = 1, \cdots, K \tag{4.19d}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{4.19e}$$

$$w_i^k \geq 0 \qquad \forall i \in V, \; \forall k = 1, \cdots, K \tag{4.19f}$$

Note that the number of variables and constraints can be huge even for a small number (around 100) of scenarios.

4.4.4 Discussion

Dealing with a large number of wind scenarios is a must in practical cases. As a consequence, the third model is rather impractical, as its solution cannot be attempted without resorting to sophisticated (and time-consuming) decomposition techniques. On the other hand, the first model is unrealistically simple as interference between turbines is neglected completely.

Therefore we decided to stick to the second model, because it is quite standard and well understood by practitioners, and, as already mentioned, a suitable definition of the input data allows one to easily take different wind scenarios into account.

However, even the second model turns out to be practically useful when just few possible turbine positions ($n \approx 100$) are involved. When n is larger, instead, this model becomes hopeless because of the huge number of variables and constraints that grows quadratically with the number of possible positions. In the next section we will propose a new model to deal with large-scale instances.

4.5 A new model

In our fourth model the so-called "Glover's trick" [13] is used, based on the use of a "big M" to generate fewer (though weaker) cuts. This means that the relaxed solution at each node is less close to the optimal integer solution, and many more tree nodes could be developed in the Branch&Bound, but the solution of each node is much faster. In a forthcoming computational section, indeed, we will show that this new model performs much better than the others when the number of possible positions is large.

4.5.1 Fourth model

Our new model is a restatement of the second model of Section 4.3.2. Variables x_i are the same and the aim is still to maximize the power production while considering all pairwise interferences,

i.e.,

$$\text{maximize} \sum_{i \in V} P_i x_i - \sum_{i \in V} \sum_{j \in V} I_{i,j} x_i x_j \tag{4.20}$$

Defining

$$w_i = \begin{cases} \sum_{j \in V} I_{i,j} x_j & \text{if } x_i = 1 \\ 0 & \text{if } x_i = 0 \end{cases}$$

the objective function (4.20) can be rewritten as

$$\text{maximize} \sum_{i \in V} (P_i x_i - w_i) \tag{4.21}$$

and the model becomes

$$\begin{aligned}
&\text{maximize} \sum_{i \in V} (P_i x_i - w_i) && && (4.22a)\\
&\text{subject to} \\
&\quad \sum_{i \in V} x_i \leq N_{MAX} && && (4.22b)\\
&\quad x_i + x_j \leq 1 && \forall i, j \in V : \; j > i, \; \text{dist}(i,j) < D_{MIN} && (4.22c)\\
&\quad \sum_{j \in V} I_{i,j} x_j \leq w_i + M_i (1 - x_i) && \forall i \in V && (4.22d)\\
&\quad x_i \in \{0, 1\} && \forall i \in V && (4.22e)\\
&\quad w_i \geq 0 && \forall i \in V && (4.22f)
\end{aligned}$$

where $M_i >> 0$ is a "big M", i.e. a very large positive number.

Note the role played by the big coefficient M_i: when $x_i = 0$ constraint (4.22d) becomes $\sum_{j \in V} I_{i,j} x_j \leq w_i + M_i$ and value $w_i = 0$ is allowed. When $x_i = 1$, instead, the constraint becomes active and imposes $\sum_{j \in V} I_{i,j} x_j \leq w_i$ as required.

The new model only uses $O(|V|)$ variables and $O(|V|)$ constraints (except (4.22c) of course), so it is suitable for large-scale instances. To our knowledge, no similar reformulation can be written for the third model.

4.5.2 Discussion

The new model is expected to give a worse approximation than the second model, due to the presence of the big M coefficients that lead to weaker constraints. On the other hand, the new model has much fewer constraints and variables, so it appears a viable option for medium to large instances, as those arising in practical applications.

We next report some illustrative results comparing the two models (two and four) on randomly generated offshore instances with $n = 50, 100, 150, 200, 300, 500, 1000$. It should be stressed that

the two models, though very different, do in fact express the same optimization problem, so the optimal solution produced by the two is the same. All results refer to the use of the state-of-the-art MILP solver IBM ILOG CPLEX 12.5.1 [6] on an PC with 2.3GHz Intel Core i7 processor and 16GB RAM (8 thread runs) with a time limit of 3600 sec.s. Table 4.1 reports, for each value of n, the main performance figures of the runs, namely:

- the number of variables, constraints, and nonzeros elements in the model matrix after the initial preprocessing; note that the number of nonzeros plays a relevant role in LP optimization time.
- the time at root node to solve the initial LP (before cuts)
- the final time spent on the root node (after the addition of cuts)
- the upper bound (optimistic estimate of the optimal solution value) available at the end of the root node (the lower the better)
- the value of the best-known solution available at the end of the root node (the larger the better)
- the number of cuts applied at the root node
- the total number of branching nodes at termination
- the best upper bound (optimistic estimate of the optimal solution value) available at termination (the lower the better)
- the value of the best-known solution available at termination (the larger the better)
- the total computing time, in wall clock seconds, at termination; 3600 means that the time limit was hit, so the current best solution is not a provable optimal solution.

	$n = 50$		$n = 100$		$n = 150$		$n = 200$		$n = 300$		$n = 500$		$n = 1000$	
	Mod 2	Mod 4	Mod 2	Mod 4	Mod 2	Mod 4	Mod 2	Mod 4	Mod 2	Mod 4	Mod 2	Mod 4	Mod 2	Mod 4
n. variables	1419	18	4786	200	9631	300	16403	400	32669	600	92257	1000	381983	2000
n. constraints	1369	9	4866	280	9608	277	16376	373	32704	635	92173	916	382420	2437
n. nonzeros	4107	90	14418	5246	28950	10288	49300	17294	99451	35313	279012	96498	1166942	406976
root LP time	0.01	0.01	0.05	0.01	0.02	0.02	0.04	0.07	20.05	0.07	26.37	0.36	380.91	4.32
root time	0.06	0.03	78.6	0.29	22.79	0.35	42.38	0.82	299.40	2.63	368.29	3.73	3600	25.97
root bound	53.63	53.63	61.19	70.12	66.05	66.96	87.55	88.69	73.30	76.88	105.09	108.26	112.40	112.78
root heu.sol.	53.63	53.63	54.38	53.1	53.46	53.96	63.50	63.84	54.67	56.13	70.24	70.78	65.64	65.64
root cuts	16	9	4960	37	1073	23	2173	11	1038	290	1040	15	178	40
final n. nodes	0	0	16K	16M	178K	27M	33K	12M	16K	3M	1778	2M	0	129K
final best bound	53.63	53.63	54.49	54.49	57.47	56.66	78.98	79.16	67.98	68.53	102.68	100.52	112.4	109.96
final best heu.sol.	53.63	53.63	54.49	54.49	54.15	54.36	65.09	66.28	55.66	56.67	70.72	74.19	65.64	71.31
final time	0.06	0.03	2006.05	1698.70	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600

Table 4.1. Comparing the second and the fourth models for different values of n

According to the table, both the second and the fourth models can find a proven optimal solution for $n = 50$ in less than one seconds, and for $n = 100$ in about half an hour. On the other hand, none of the two can complete the enumeration for $n \geq 150$ within one hour. As expected, model four produces a weaker (i.e. larger) upper bound than the second model at the root node. However, model four is much faster than model two in processing each branching node, which results into a much larger tree be explored within the time limit. As to the value of the best solution available at termination, model four is the clear winner, especially for $n \geq 500$. Note that for $n \geq 1000$ the second model turns out to be just impractical, because even the root node computation cannot be completed within one hour. Model four, instead, has much fewer variables and constraints so it can be processed much more efficiently.

As our goal is to address much larger instances with $n = 10000+$, we cannot expect to be able to use the second model. To this purpose, neither the fourth model can be applied as it is, but

needs to be embedded into a heuristic approach in which the model is used to refine a given heuristic solution. In this setting, model four was preferred because it seems more powerful in producing fast heuristic solutions. The developed of such an approach will be addressed in the next chapter.

5 Solution Methods

The aim of this chapter is to heuristically solve instances with 10000+ possible positions, within a few minutes on a standard PC. Our fourth model in Chapter 4 is used as it is the only feasible option for large instances. The solution of this model is speeded up by the use of some heuristics, namely 1-opt, 2-opt and Proximity Search.

5.1 Ad hoc heuristics: 1-opt and 2-opt

The 2-opt strategy has been developed in 1958 to solve the Traveling Salesman Problem [14]. The idea is to evaluate every possible pair of edge flips and see if the solution improves. This technique can be generalized to k-opt where all the possible combinations of k flips are considered. In our case we implemented a 1-opt and a 2-opt strategy to construct a first solution that the MILP solver will try to improve.

We will next describe our 1-opt heuristic, while the 2-opt one is not described as it is a simple extension of 1-opt. The idea of our 1-opt heuristic is to search, step by step, a better solution by flipping only one variable x_i at time. At each iteration, the incumbent solution $\tilde{x}$ is stored, i.e., $\tilde{x}$ describes the best-known solution so far (as usual, $\tilde{x_i} = 1$ if a turbine is built in position i, $= 0$ otherwise). The current solution is instead denoted by x. For the sake of generality, we also consider a minimum number N_{MIN} of turbines to be built.

At each step we explicitly store the profit for the current solution x

$$z = \sum_{i \in V} P_i x_i - \sum_{i \in V} \sum_{j \in V} I_{i,j} x_i x_j \tag{5.1}$$

and its cardinality

$$\gamma = \sum_{i \in V} x_i \tag{5.2}$$

When the 1-opt algorithm flips a variable x_j, the profit of exchanges by a quantity δ_j equal to

$$\delta_j = \begin{cases} P_j - \sum_{i \in V : x_i = 1} (I_{i,j} + I_{j,i}) & \text{if } x_j = 0 \\ -P_j + \sum_{i \in V : x_i = 1} (I_{i,j} + I_{j,i}) & \text{if } x_j = 1 \end{cases} \qquad (j \in V)$$

where we assume $I_{i,j} = BIGM$ for all sites i, j that are incompatible because of the proximity constraint ($BIGM$ being a large penalty, for example $BIGM = \sum_{i \in V} P_i$), while $I_{i,i} = 0$ as usual.

The algorithm is now taking into account the interference and the proximity constraints, but not the limit on the number of turbines. We therefore extended the algorithm as follows. The

idea is to improve the current x by looking in $O(n)$ time for the site j^* such that

$$j^* = argmax_{j \in V}\{\delta_j + FLIP(j)\} \tag{5.3}$$

The quantity $FLIP(j)$ is used to take into account the constraint related to the min/max number of turbines. Indeed, $FLIP(j)$ is defined as:

$$FLIP(j) = \begin{cases} -HUGEM & \text{if } x_j = 0 \text{ and } \gamma \geq n_2 \\ -HUGEM & \text{if } x_j = 1 \text{ and } \gamma \leq n_1 \\ +HUGEM & \text{if } x_j = 0 \text{ and } \gamma < n_1 \\ +HUGEM & \text{if } x_j = 1 \text{ and } \gamma > n_2 \\ 0 & \text{otherwise} \end{cases}$$

where n_1 is a local copy of N_{MIN}, n_2 is a local copy of N_{MAX}, $HUGEM$ is a huge value bigger than $BIGM$. Note the role of $HUGEM$ in the maximization (5.3):

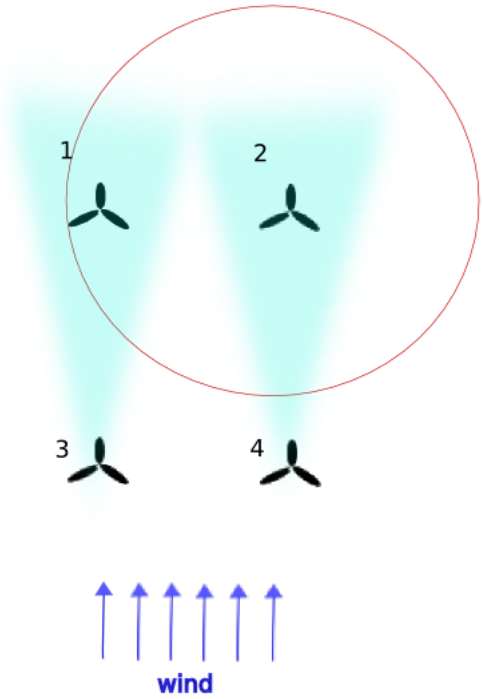
- if the cardinality γ of the current solution is greater than or equal to the maximum number of turbines allowed, term $-HUGEM$ penalizes the addition of a new turbine to the solution
- if the cardinality γ of the current solution is less than or equal to the minimum number of turbines allowed, term $-HUGEM$ penalizes the removal of a turbine from the solution
- on the contrary, if the current solution has cardinality γ less than the minimum number of turbines, term $+HUGEM$ forces the new solution to build a new turbine
- if the current solution has cardinality γ greater than the maximum number of turbines, $+HUGEM$ forces to choose the removal of a turbine from the new solution.

Parameter $HUGEM$ has been selected such that $HUGEM >> BIGM$ so that the number of turbines is considered more important than the interference in the optimization.

The algorithm starts with $x = 0, \gamma = 0, z = 0, n_1 = N_{MIN}, n_2 = N_{MAX}$. Once the best j according to (5.3) , say j^*, is found, if $\delta_{j^*} + FLIP(j^*) > 0$ then x_{j^*} is flipped and x, z, γ are updated in $O(1)$ time, δ is updated in $O(n)$ time, and the loop is continued.

Example

To better understand the 1-opt algorithm, let us consider the simple example with four possible positions illustrated in Figure 5.1. The red circle shows the minimum distance between turbines: positions 1 and 2 cannot be occupied at the same time, and neither 3 and 4, due to the proximity constraint. The wind is coming from south so 3 and 4 are the upwind turbines. The blue shadows in the figure represent interference. Table 5.1 records the possible power production at each position, the possible interference suffered by a turbine in that position, and the incompatibilities due to proximity.



position	power	interference	incompatible with
1	$P_1 = 5$	$I_{3,1}$=2	2
2	$P_2 = 10$	$I_{4,2}$=1	1
3	$P_3 = 7$	–	4
4	$P_4 = 4$	–	3

Figure 5.1. Example case

Consider $n_1 = N_{MIN}$=2 and $n_2 = N_{MAX}$=3.

Figure 5.2 illustrates an iteration starting from $x = (0, 1, 0, 0)$. In the figure, $\delta_j + FLIP(j)$ is explicitly computed in every block.

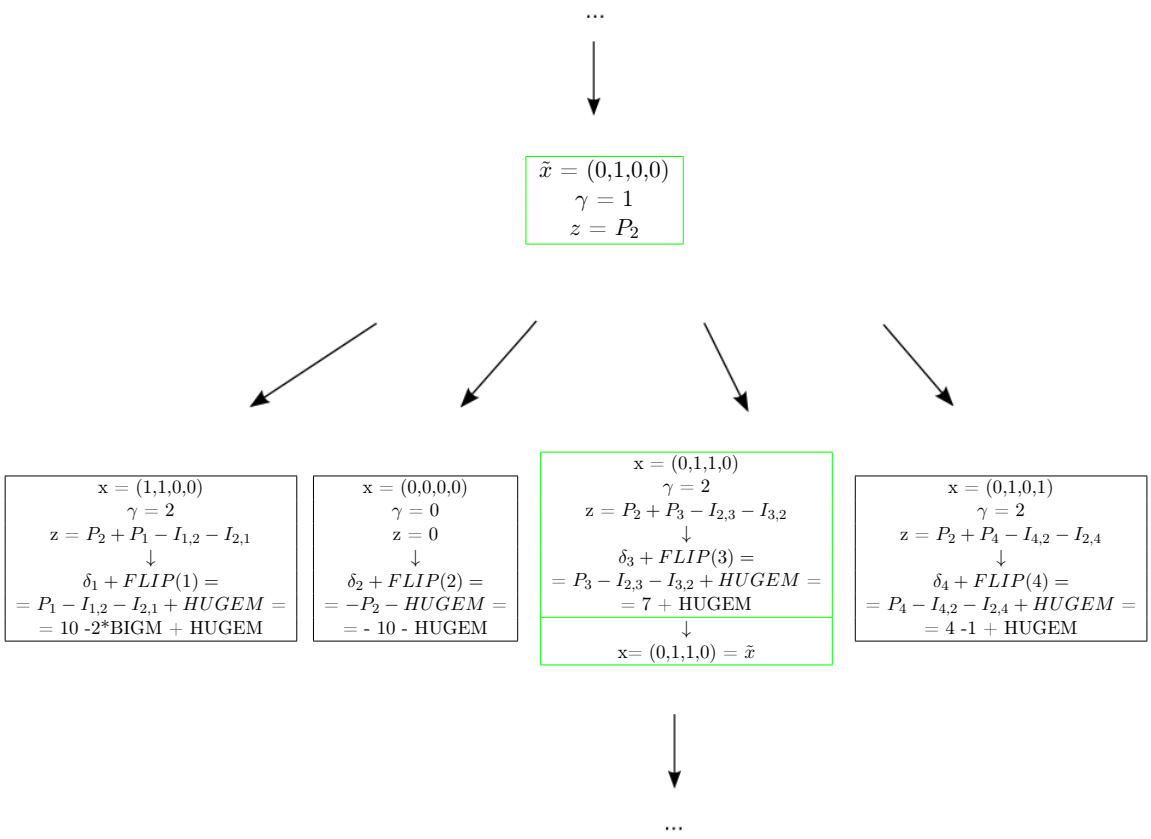


Figure 5.2. An illustrative example of the 1-opt heuristic

In the example j^* corresponds to the third block (the green one) so $\tilde{x}$ is updated to (0,1,1,0) and the algorithm will proceed from that solution.

The algorithm continues until it reaches a local minimum, i.e., $\delta_{j^*} + FLIP(j^*) \leq 0$, meaning that the current solution x cannot be improved by flipping only one x_j. To escape from this local minimum, different metaheuristics could be applied as, e.g., Tabu Search [15] or Variable Neighborhhod Search [16]. In our implementation, however, we implemented a simpler scheme that produced good results for our instances. The idea is to modify the values n_1 and n_2 to force solution x to move from the local minimum. In particular, values n_1 and n_2 are updated as follow:

$$n_1 := n_2 := \begin{cases} \gamma + \rho\gamma/2 + 10 & \text{if } \gamma \leq \sum_{i \in V} \tilde{x}_i \\ \gamma - \rho\gamma/2 - 10 & \text{otherwise} \end{cases}$$

where ρ is a uniformly pseudo-random value in $[0, 1]$.

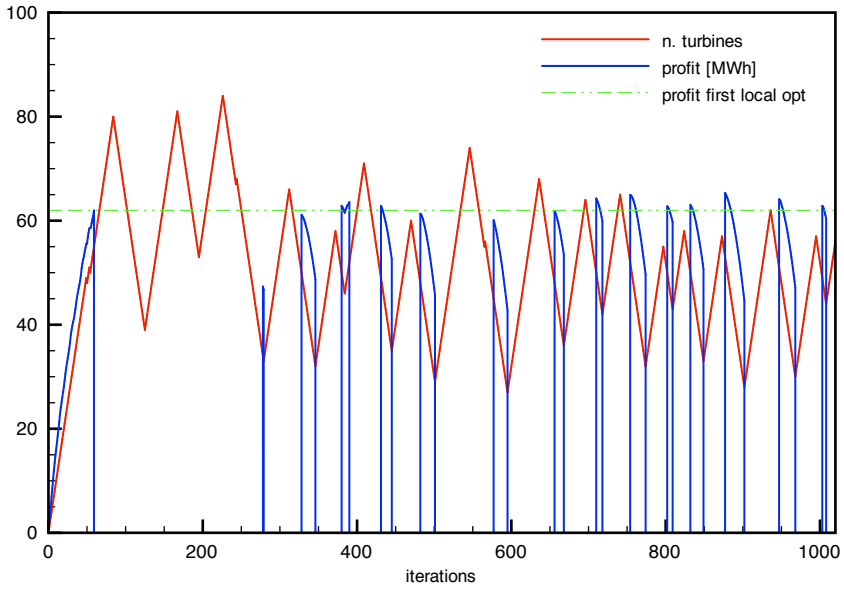


Figure 5.3. Plot of the number of turbines and profit of the current solution during the heuristic algorithm

Figure 5.3 illustrates the oscillatory behavior of our heuristic on a sample problem with 1000 possible positions on an offshore grid. The first iterations produce a sequence of improved solutions obtained by adding one turbine at time, till a local optimum of value 61.97 with 55 turbines at iteration 60 (horizontal green line in the figure). To escape form this local optimum, our heuristic changes n_1 and n_2 to 80, in order to force the solution to install additional turbines, even if the profit reduces due to interference, and actually becomes very negative because of the -BIGM contribution due to the violation of the proximity constraint. Afterwards, $n_1 = n_2$ go up and down and force the current solution to change (see the red line zig-zagging in the figure). From time to time, the current solution becomes feasible again (i.e. with a positive profit) and eventually improves the previous best solution. In the figure this happens around iterations 400, 700 and 900.

5.2 MILP based heuristics

5.2.1 Proximity Search

The Proximity Search [17] approach has been used to improve the best-known solution $\tilde{x}$ by using a black-box MILP based on our fourth model.
The idea is to use the MILP solver on a slightly modified problem that models the search of an improved solution in a neighborhood of $\tilde{x}$. This is done by modifying the model as follows:

Original model

$$\text{maximize} \sum_{i \in V} (P_i x_i - w_i) \tag{5.4a}$$

subject to

$$N_{MIN} \leq \sum_{i \in V} x_i \leq N_{MAX} \tag{5.4b}$$

$$x_i + x_j \leq 1 \qquad \forall i, j \in V : \; j > i, \; \text{dist}(i,j) < D_{MIN} \tag{5.4c}$$

$$\sum_{j \in V} I_{i,j} x_j \leq w_i + M_i (1 - x_i) \qquad \forall i, j \in V \tag{5.4d}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{5.4e}$$

$$w_i \geq 0 \qquad \forall i \in V \tag{5.4f}$$

Modified model 4 (for Proximity search)

$$\text{minimize} \sum_{i \in V : \tilde{x_i} = 0} x_i + \sum_{i \in V : \tilde{x_i} = 1} (1 - x_i) \tag{5.5a}$$

subject to

$$N_{MIN} \leq \sum_{i \in V} x_i \leq N_{MAX} \tag{5.5b}$$

$$x_i + x_j \leq 1 \qquad \forall i, j \in V : \; j > i, \; \text{dist}(i,j) < D_{MIN} \tag{5.5c}$$

$$\sum_{j \in V} I_{i,j} x_j \leq w_i + M_i (1 - x_i) \qquad \forall i, j \in V \tag{5.5d}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{5.5e}$$

$$w_i \geq 0 \qquad \forall i \in V \tag{5.5f}$$

$$\sum_{i \in V} (P_i x_i - w_i) \geq \sum_{i \in V} (P_i \tilde{x_i} - \tilde{w_i}) + \theta \tag{5.5g}$$

where $\theta > 0$ is a given tolerance, and $\sum_{i \in V} (P_i \tilde{x_i} - \tilde{w_i})$ represents the profit of the current solution $\tilde{x}$.

The new objective function minimizes the Hamming distance between the new x and $\tilde{x}$. In other words, the new x should be as similar as possible to $\tilde{x}$ (we are exploring its neighborhood) but at the same time has to improve its profit by a given quantity θ according to constraint (5.5g). In our implementation, we used an improved version of the above scheme, called "proximity search with an incumbent" in [17]. It consists of introducing a continuous variable $\xi \geq 0$ and in weakening (5.5g) to

$$\sum_{i \in V} (P_i x_i - w_i) \geq \sum_{i \in V} (P_i \tilde{x_i} - \tilde{w_i}) + \theta (1 - \xi)$$

while minimizing $\sum_{i \in V : \tilde{x_i} = 0} x_i + \sum_{i \in V : \tilde{x_i} = 1} (1 - x_i) + M\xi$, where $M >> 0$ is a very large value (see [17] for details).

Computational experience reported in [17] shows that Proximity Search is quite successful (in particular, on some classes of problems involving big-M constraint), due to the action of the proximity objective function that is beneficial both in speeding up the solution of the LP relaxations, and in driving the heuristics embedded in the MILP solvers.

5.3 Our overall heuristic

Our final heuristic is a mixture of 1-opt and 2-opt heuristics and MILP refining based on proximity search, and is illustrated in Figure 5.4.

Step 0. read input data and compute the overall interference matrix (I_{ij});
Step 1. apply the 1-opt heuristic to get a first incumbent $\tilde{x}$;
Step 2. apply quick ad-hoc refinement heuristics (few iterations of iterated 1- and 2-opt) to possibly improve $\tilde{x}$;
Step 3. if $n > 2000$, randomly remove points $i \in V$ with $\tilde{x}_i = 0$ so as to reduce the number of candidate sites to 2000;
Step 4. build a MILP model for the resulting subproblem and apply proximity search to refine $\tilde{x}$ until the very first improved solution is found (or time limit is reached);
Step 5. if time limit permits, repeat from Step 2.

Figure 5.4. Our overall heuristic framework

At Steps 1 and 2 the ad-hoc heuristics of Section 5.1 are applied.

Two different MILP models are used to feed the proximity-search heuristic at Step 4. During the first part of the computation, we use a simplified MILP model obtained from (5.5a) by removing all interference constraints (5.5d), thus obtaining a much easier relaxation. A short time limit (60 sec.s) is imposed for each call of proximity search when this simplified model is solved. In this way we aggressively drive the solution $\tilde{x}$ to increase the number of built turbines, without being bothered by interference considerations and only taking pairwise incompatibility (5.5c) into account. This approach quickly finds better and better solutions (even in terms of the true profit), until either (i) no additional turbine can be built, or (ii) the addition of new turbines does in fact reduce the true profit associated to the new solution. In this situation we switch to the complete model (5.5a) with all interference constraints, which is used in all next executions of Step 4. Note that the simplified model is only used at Step 4, while all other steps of the procedure always use the true objective function that takes interference into full account.

6 Computational tests

In this section different tests have been performed by considering different wind scenarios and number of sites. Wind scenarios used to compute the interference matrix are taken from the European Wind Energy Association (EWEA) data, aggregated in bins according to 24 wind-angles (15 degrees each) and wind speeds in 1m/s per bin.

6.1 Small test case

Our fourth model was first tested on an offshore grid of 10×10 possible positions in a square of $3000 \times 3000m$, with D_{MIN} 400 m.

The first set of plots (Figure 6.1) shows how the optimal layout changes when imposing a different maximum number of turbines. Red circles represent the built turbines, while the color in the background refers to the interference induced by those turbines.

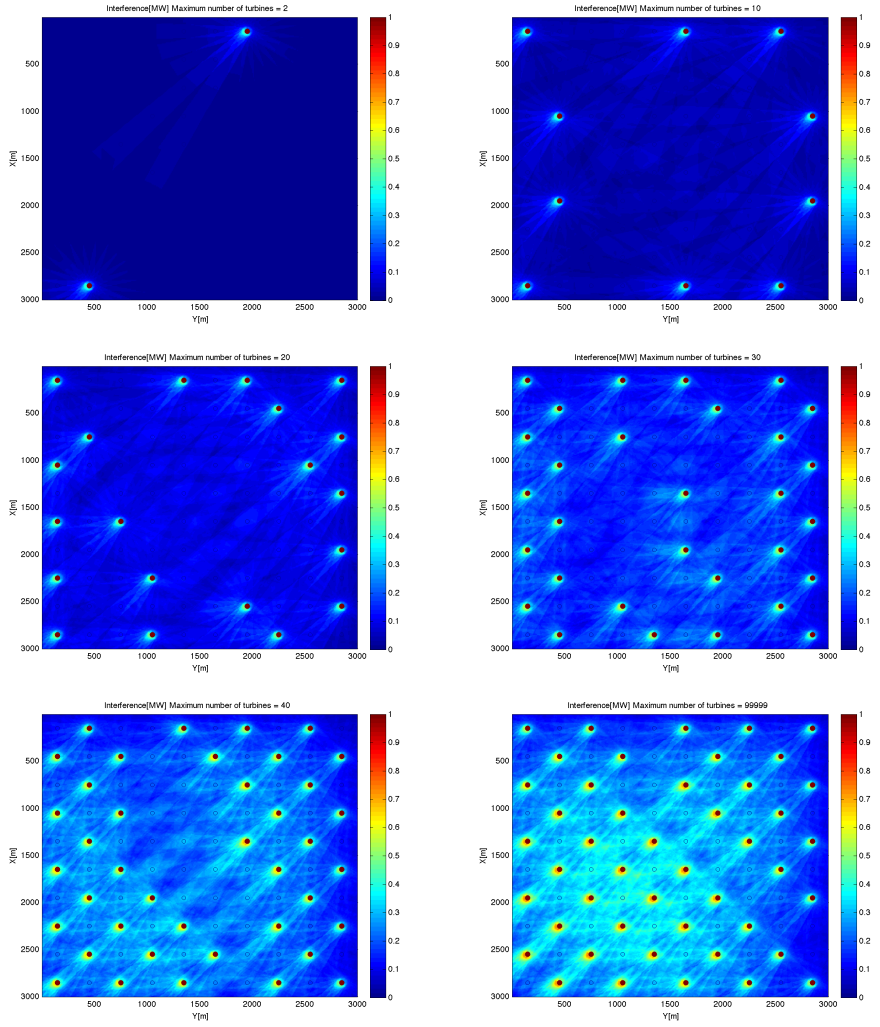


Figure 6.1. Optimal layout solutions imposing maximum number of turbine equal to 2, 10, 20, 30, 40, and unlimited

Note from the last plot that not all the points in the grid can be occupied due to the proximity constraint (the distance between the grid points is 300m, while the minimum distance imposed is 400m). It interesting to notice that the optimal solutions have as many turbines as possible on the boundaries: those are, indeed, the positions in which a turbine would cause less interference to the others.

The second set of plots (Figure 6.2) shows how the optimal layout changes when considering different wind scenarios. An increasing number of scenarios from EWEA data has been considered when computing the average interference, the last plot referring to all the 250 000+ real-world wind samplings of the EWEA data grouped into about 500 macro-scenarios.

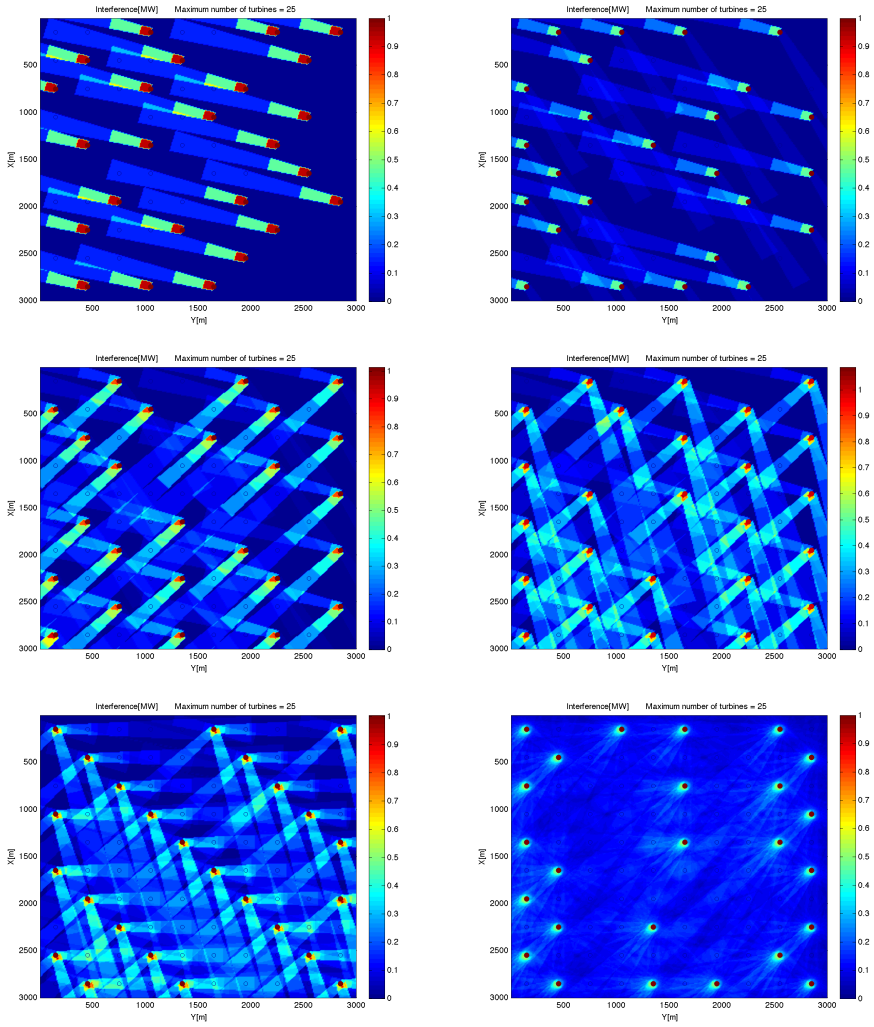


Figure 6.2. Optimal layout solutions considering a maximum number of turbines equal to 25, and 1, 2, 3, 4, 10, 500 wind scenarios

It can be noticed that, for a same maximum number of turbines imposed, the optimal solution changes due to interference. As expected, turbines are allocated so as to minimize their pairwise interference.

6.2 A real-world onshore case

As we have seen in Chapter 2, in the onshore problem many constraints are involved in the selection of the possible turbines positions.

We considered the real-world test case described in Chapter 2, referring to a real site in United Kingdom. The allowed positions turn out to identify 3103 potential points (blue area in the last plot in Figure 2.8).

A feasible layout with unlimited number of turbines for our site is shown in the figure below.

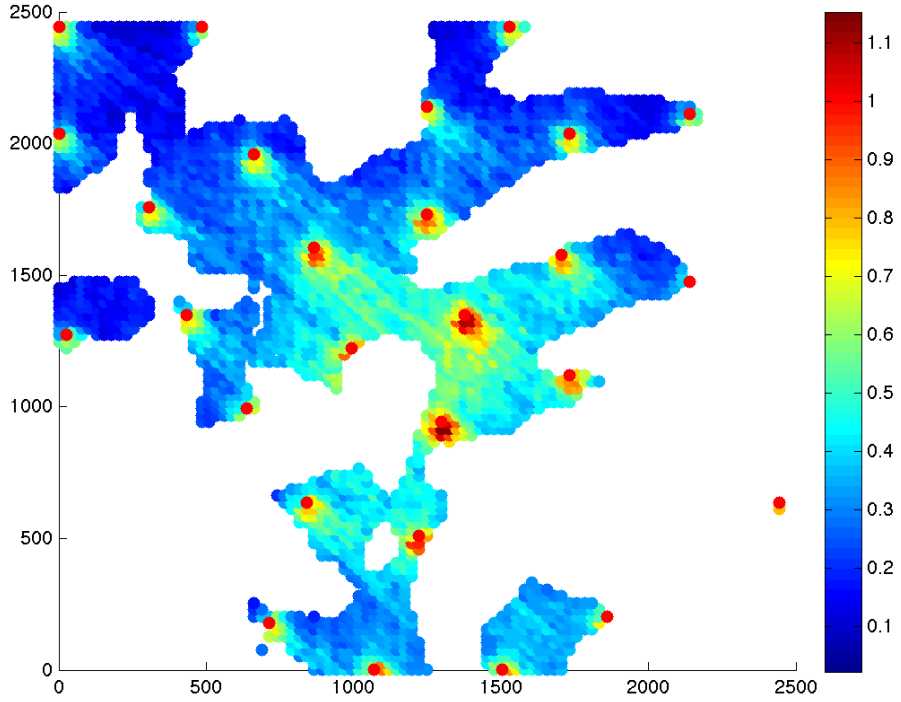


Figure 6.3. Feasible layout with 27 turbines; colors refer to interference computed as explained in Chapter 3

Another key point in the onshore scenario is that the wind is not homogeneously distributed, i.e., a turbine can give different power productions depending on its position (even without considering any interference). With respect to the possible positions obtained by the previous analysis (first plot in Figure 6.4), the actual power distribution is shown in the second plot of Figure 6.4.

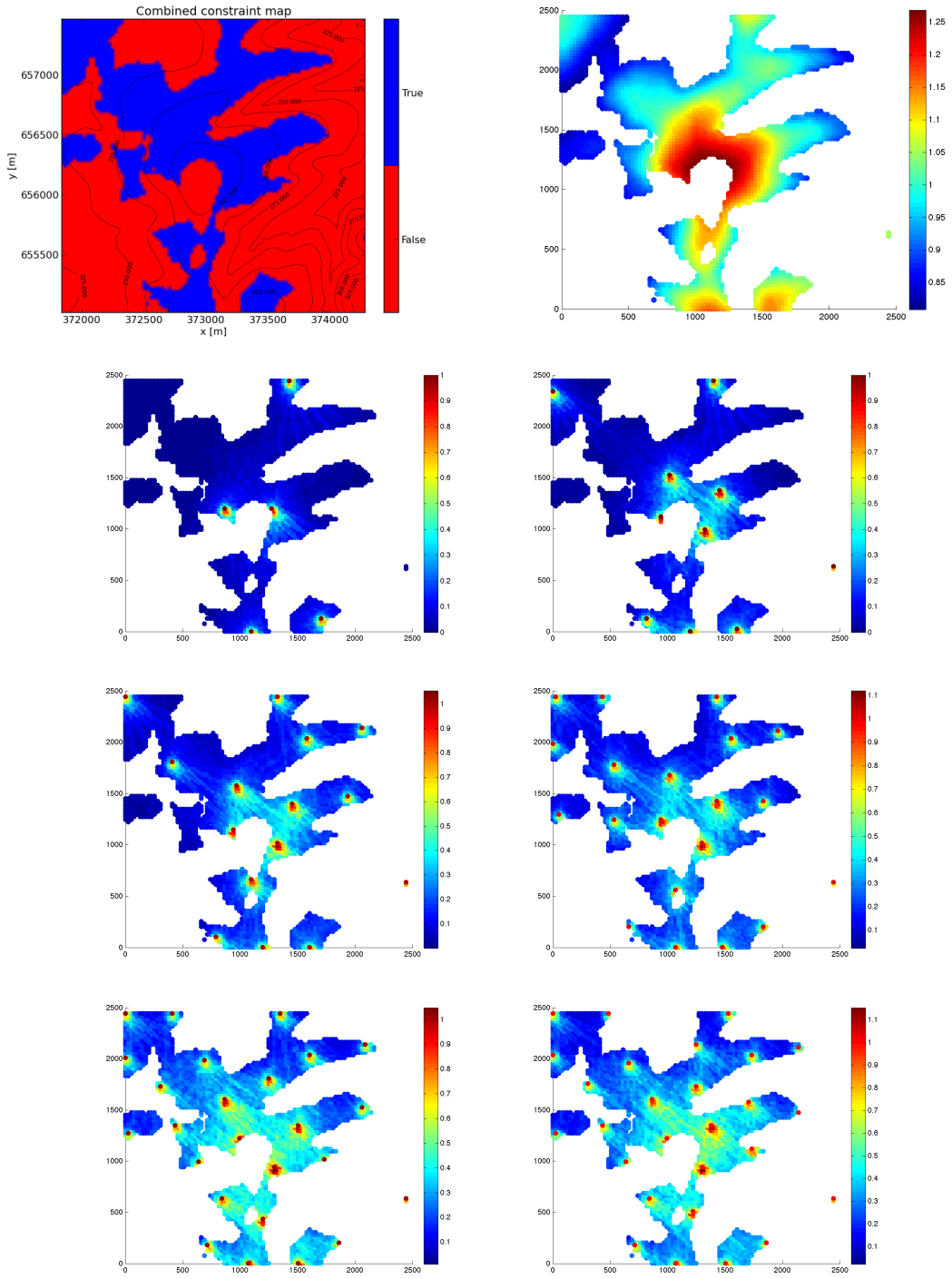


Figure 6.4. The first plot shows feasible positions while the second shows the power distribution within the site. All other plots show our layout solutions when considering a maximum number of turbines equal to 5, 10, 15, 20, 25, and unlimited, respectively

6.3 A large-scale real-world offshore case

In this section we address a real-world offshore case from EWEA data. Our heuristic approach presented in Chapter 5 is computationally compared with alternative approaches. The tests are designed to compare the performance of different methods using the same MILP solver – in our case, IBM ILOG Cplex 12.5.1 [6] – on very large-scale instances (up to 20 000 possible positions).

Interference has been computed by considering 250 000+ real-world wind samples for the EWEA site, grouped into about 500 macro-scenarios. As to possible turbine positions within the real-world site, in the offshore case they are typically sampled on a regular grid. However, to have more instances to compare for the given single site, we preferred to generate them randomly according to a uniform distribution. In this way we are able to compute meaningful performance figures for the algorithms under comparison. Clearly, the more possible positions, the better the final solution, so large-scale problems come into play.

Our comparison is made among the following alternative solution methods:

a) `proxy`: this approach is the one presented in Chapter 5. To improve performance, some parameters of Cplex have been tuned in an aggressive way: all cuts deactivated, `CPX_PARAM_RINSHEUR` = 1, `CPX_PARAM_POLISHAFTERTIME` = 0.0, `CPX_PARAM_INTSOLLIM` = 2;

b) `cpx_def`: the application of solver Cplex in its default setting, aimed to find a proven optimal solution. The model given to the solver is the fourth one, presented in Chapter 4. As the `proxy` approach is starting from the heuristic solution $\tilde{x}$ found after the first execution of 1-and 2-opt, also `cpx_def` is starting from the same solution, so the comparison is fair;

c) `cpx_heu`: as our problems involve a huge number of variables, it is hopeless to find an optimal solution in a reasonable time limit. Therefore we considered the `cpx_heu` approach, i.e., the same approach as `cpx_def` but using the following aggressive tuning to improve the heuristic performance: all cuts deactivated, `CPX_PARAM_RINSHEUR` = 100, `CPX_PARAM_POLISHAFTERTIME` = 20% of the total time limit of 3600sec.s;

d) `loc_sea`: a simple local-search procedure not based on any MIP solver, that just uses 1-opt and 2-opt and randomly removes installed turbines from the current best solution after 10,000 iterations without improvement of the incumbent.

In our view, `loc_sea` is representative of a clever but not oversophisticated metaheuristic, as typically implemented in practice, while `cpx_def` and `cpx_heu` represent a standard way of exploiting a MILP model once a good feasible solution is known.

Our testbed refers to an offshore 3000 × 3000 (m) square with $D_{MIN} = 400$ (m) minimum turbine separation, with no limit on the number of turbines to be built (i.e., $N_{MAX} = +\infty$).

We generated five classes of medium-to-large problems with $n = 1000$, 5000, 10000, 15000, and 20000 possible positions. For each class, 10 instances have been considered by generating n uniformly random points in the 3000 × 3000 square as possible turbine positions. In what follows, reported computing times are in CPU sec.s of an Intel Xeon E3-1220 V2 quad-core PC with 16GB of RAM, and do not take interference matrix computation into account as this matrix is assumed to be precomputed and reused at each run. In our experiments, a time limit of 3600 seconds has been imposed, which is a reasonable limit for what-if analyses typically performed by practitioners.

Computational results on our instances are given in Table 6.1, where each entry refers to the performance of a given algorithm at a given time limit. In particular, the left part of the table reports, for each algorithm and time limit, the *number of wins*, i.e., the number of instances for which a certain algorithm produced the best solution at the given time limit (ties included).

According to the table, `proxy` outperforms all competitors by a large amount for medium to large instances. As expected, `cpx_heu` performs better for instances with $n = 1000$ as it is allowed to explore a large number of enumeration nodes for the original model and objective function. Note that `loc_sea` has a good performance for short time limits and/or for large instances, thus confirming its effectiveness, whereas `cpx_heu` is significantly better than `loc_sea` only for small instances and large time limits.

A different performance measure is given in the right-hand side part of Table 6.1, where each entry gives the *average optimality ratio*, i.e., the average value of the ratio between the solution produced by an algorithm (on a given instance at a given time limit) and the best solution known for that instance—the closer to one the better. It should be observed that an improvement of just 1% has a very significant economical impact due to the very large profits involved in the wind farm context. The results show that `proxy` is always able to produce solutions that are quite close to the best one. As before, `loc_sea` is competitive for large instances when a very small computing time is allowed, whereas `cpx_def` and `cpx_heu` exhibit a good performance only for small instances, and are dominated even by `loc_sea` for larger ones.

Figure 6.5 plots the incumbent value (i.e., the profit of the current best solution) over CPU time for the four heuristics under comparison, and refer to 4 sample instances in our testbed. The two subfigures on the top refer to two small instances with $n = 1000$, where `proxy`, `cpx_heu` and `cpx_def` have a comparable performance and clearly outperform `loc_sea`. For $n = 5000$ (bottom-left subfigure) and $n = 10000$ (bottom-right subfigure), however, both `cpx_def` and `cpx_heu` (and also `loc_sea`) have hard time in improving their initial solution, and are outperformed by `proxy` by a large amount.

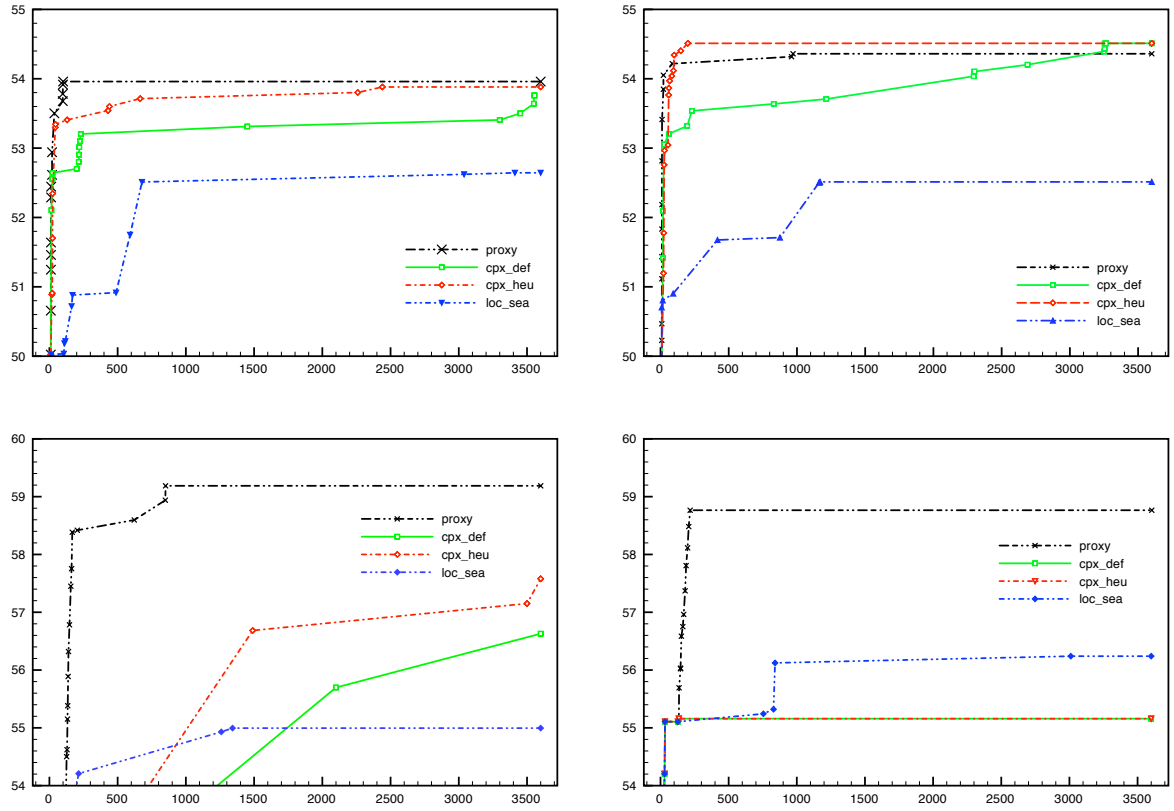


Figure 6.5. Solution profit over time for 4 sample instances with $n = 1000$ (top left and top right), $n = 5000$ (bottom left), and $n = 10000$ (bottom right); the higher the profit the better.

Table 6.1. Number of times each algorithm finds the best solution within the time limit (wins), and optimality ratio with respect to the best known solution—the larger the better.

		number of wins				optimality ratio			
n	Time limit (s)	`proxy`	`cpx_def`	`cpx_heu`	`loc_sea`	`proxy`	`cpx_def`	`cpx_heu`	`loc_sea`
1000	60	6	1	3	0	0.994	0.983	0.987	0.916
	300	4	2	4	0	0.997	0.991	0.998	0.922
	600	7	3	7	0	0.997	0.992	0.997	0.932
	900	5	2	3	0	0.998	0.993	0.996	0.935
	1,200	5	1	5	0	0.998	0.992	0.997	0.939
	1,800	5	1	4	0	0.998	0.992	0.996	0.942
	3,600	4	2	5	0	0.998	0.995	0.997	0.943
5000	60	9	6	6	5	0.909	0.901	0.901	0.904
	300	10	0	0	0	0.992	0.908	0.908	0.925
	600	10	0	10	0	0.994	0.908	0.994	0.935
	900	10	0	0	0	0.994	0.908	0.908	0.936
	1,200	10	0	0	0	0.994	0.908	0.925	0.939
	1,800	9	0	1	0	0.996	0.908	0.971	0.946
	3,600	5	0	5	0	0.996	0.932	0.994	0.948
10000	60	9	9	8	10	0.914	0.913	0.914	0.914
	300	10	2	2	2	0.967	0.927	0.927	0.936
	600	10	0	10	0	0.998	0.928	0.998	0.944
	900	10	0	0	0	1.000	0.928	0.928	0.948
	1,200	10	0	0	0	1.000	0.928	0.928	0.951
	1,800	10	0	0	0	1.000	0.928	0.928	0.957
	3,600	9	0	0	1	1.000	0.928	0.928	0.964
15000	60	9	10	9	9	0.909	0.912	0.911	0.909
	300	10	8	7	8	0.943	0.937	0.935	0.937
	600	10	0	10	0	0.992	0.939	0.992	0.942
	900	10	0	0	0	1.000	0.939	0.939	0.956
	1,200	9	0	0	1	1.000	0.939	0.939	0.959
	1,800	9	0	0	1	1.000	0.939	0.939	0.965
	3,600	9	0	0	1	1.000	0.939	0.939	0.972
20000	60	9	9	9	10	0.901	0.902	0.901	0.902
	300	10	8	10	10	0.933	0.933	0.933	0.933
	600	9	0	9	1	0.956	0.935	0.956	0.941
	900	10	0	0	0	0.978	0.935	0.935	0.945
	1,200	10	0	0	0	0.991	0.935	0.935	0.950
	1,800	10	0	0	0	0.999	0.935	0.935	0.963
	3,600	10	0	0	0	1.000	0.935	0.935	0.971
ALL	60	42	35	35	34	0.925	0.922	0.922	0.909
	300	44	20	23	20	0.966	0.939	0.940	0.930
	600	46	3	46	1	0.987	0.941	0.987	0.938
	900	45	2	3	0	0.994	0.941	0.941	0.944
	1,200	44	1	5	1	0.997	0.940	0.945	0.947
	1,800	43	1	5	1	0.999	0.940	0.954	0.955
	3,600	36	2	10	2	0.999	0.946	0.959	0.959

7 Validation

To understand if our optimization tool is competitive with a commercial standard, a comparison with the *Optimize* module of *WindPRO version 2.9.207* was made. This software is used nowadays in many companies all over the world, including Vattenfall where we have performed our comparison tests. The tests are carried out on the real-world onshore site introduced in Chapter 2.

7.1 WindPRO's Optimize

WindPRO is a Windows 2000/XP/Vista software for the design and planning of single wind turbines and of wind farms [9]. It includes an optimization module, called *Optimize*, able to design a wind farm layout for a specific site according to energy resources, imposing a maximum/minimum number of turbines and a minimum distance between turbines. The optimization approach is a greedy heuristic (as we will explain below) that does not include any mathematical programming formulation. For more information please refer to [9].

In *WindPRO's Optimize* the user is able to select two types of optimization: *Fast energy layout* or *Full energy optimize*.

The *Fast energy layout* mode is faster than *Full energy optimize* but does not consider wake effects.

The ad-hoc heuristic applied in *Fast energy layout* works as follows: it finds the available location with the highest energy production according to the wind resource map and it places a turbine there. The next turbine will be placed on the feasible location with the highest energy resource, and so on. Optimization finishes when the maximum number of turbines is placed, or when no other turbine can be added without violating a constraint.

Full energy optimize is another ad-hoc heuristic, similar to the previous one, but modified to deal with interference between turbines. This algorithm works as follows: the first turbine is placed in the location with the highest power production, the second one in the second location with highest production feasible with the constraints. After the second turbine is placed, the first turbine is moved to find the new optimal position taking the wake of the second turbine into account. This is repeated every time a new turbine is placed, i.e. the turbines previously placed are iteratively moved to improve efficiency.

All *WindPRO* algorithms are heuristics that cannot ensure that the final layout found is an optimal one. In our validation test the comparison is done with the *Full energy optimize* module.

7.2 Modification to make profit functions comparable

The *Optimize* module of *WindPRO* finds the layout according to the annual energy production, in MWh/y. On the contrary our tool, as presented so far, is optimizing according to the average power production in MW (computed from the average wind speed). Figure 7.1 shows the difference between the two power distributions in our test site: the first power map represents the power production in MW in each point (not considering any interference). Values have been computed by considering the average wind speed computed by *WindPRO* and the power curve of Siemens SWT-2.3-93 turbine (as illustrated in Chapter 3). The second plot shows the annual energy production map (in MWh/y) as computed by *WindPRO*.

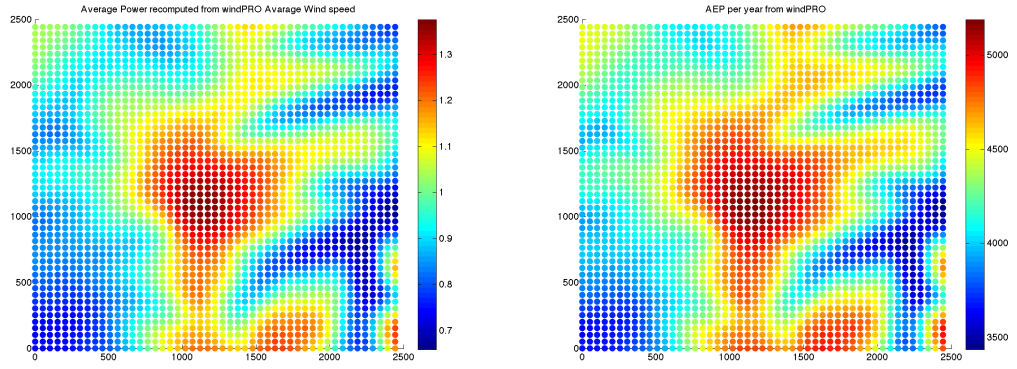


Figure 7.1. Power distribution within the site: left plot is the average power production in MW we computed from the average wind speed given by *WindPRO*, the right one is the annual energy production in MWh/y computed by *WindPRO*.

Since the two power maps are not identical, and this can change the optimal layout, our profit function has been modified to deal with annual energy production. To be sure to consider the same power, we used as power in each possible position the value uploaded from *WindPRO* (second map in Figure 7.1). The interference is computed on the power curve in MW (see Chapter 3) so it needs to be scaled in MWh/y. To do that we used a simple proportion: we computed the interference from the average wind, say δP_{AW}, as discussed in Chapter 3, and converted this value in the loss of annual energy production, say δP_{AEP}, using the formula:

$$\delta P_{AEP} = \frac{P_{AEP}\ \delta P_{AW}}{P_{AW}} \tag{7.1}$$

Where: P_{AEP} = is the annual energy production in MWh/y from *windPRO* (second plot in Figure 7.1)
δP_{AEP} = is the loss in annual energy production, in MWh/y
P_{AW} = is the average energy production, in MW, computed from *windPRO*'s average wind speed (first plot in Figure 7.1)
δP_{AW} = is the loss in energy production, in MW, computed using Jensen's model

Since *WindPRO* does not explain in details how the interference is computed in its *Optimize* module, we carried some tests comparing our solution profit with the one from *Optimize* for a

same layout. From these comparisons, it turns out that our interference should be scaled by a factor 0.5. With these little modifications, our profit function becomes comparable with the *windPRO* one, hence it will be used in all tests reported in the next section. Table 7.1 shows the two profit functions compared on five test cases, i.e five layouts with different number of turbines (5, 10, 15, 20, and 22, respectively).

N_{MAX}	WindPRO profit (MWh/y)	Our profit (MWh/y)	Difference
5	51 695	51 776	0.2%
10	99 330	99 577	0.2%
15	143 727	144 789	0.7%
20	185 285	185 252	0.0%
22	200 699	200 412	-0.1%

Table 7.1. Comparison between *Optimize* profit function (in the second column) and our modified profit function (third column) on the same layouts. Both profits are taking interference into account. The profit is computed for 5 test layouts with 5, 10, 15, 20 and 22 turbines.

7.3 Comparison

As already discussed, all tests in this section are carried on the real-world onshore site from EWEA already presented and refer to Siemens SWT-2.3-93 turbines. Both optimizers are considering the same possible positions obtained by imposing the same constraint map (the one illustrated in Chapter 6) and with the same grid resolution (50m). A minimum distance of 400m is imposed. *WindPRO's Optimize* is run in its *Full energy optimize* mode while our tool is the one using the `proxy` approach explained in details in Chapter 5. Layouts are optimized and compared according to the profit function in Section 4.5, modified as explained in Section 7.2. That means that the layouts, in both *WindPRO* and our cases, are optimized to produce as much power as possible by considering interference between turbines.

Our first test compares *WindPRO Optimize*'s layout with our layout when imposing no limit on the number of turbines. In other words, both optimizers are allowed to build as many turbines as possible in order to maximize power production. This comparison is intended to test the effectiveness of optimization algorithms. The case with unlimited number of turbines, indeed, is the most challenging for optimization.

Figure 7.2 compares the layout found by the two optimizers. The yellow area represents the feasible area, the blue dots are the turbines built by *WindPRO's Optimize*, while the red stars are those computed by our tool.

Table 7.2 shows the optimal profits given by *WindPRO's Optimize*, by our initial 1-opt and 2-opt heuristics, and by our complete tool (with proximity search) using the MILP solver. As to computing times, *WindPRO* requires minutes of computation, while our 1- and 2-opt heuristics require just a few seconds. Note that our tool is able to put 6 more turbines than *WindPRO* in the same site with the same restrictions. The second column of Table 7.2 refers to the solution of our initial local-search heuristics: our heuristics, in few seconds, are already able to locate 28 turbines, outperforming the *WindPRO's Optimize* layout by more than 15%. The MILP model is then able to move those turbines in better positions, improving power production even more (compare with the third column in Table 7.2) and outperforms *WindPRO* by more than 20%.

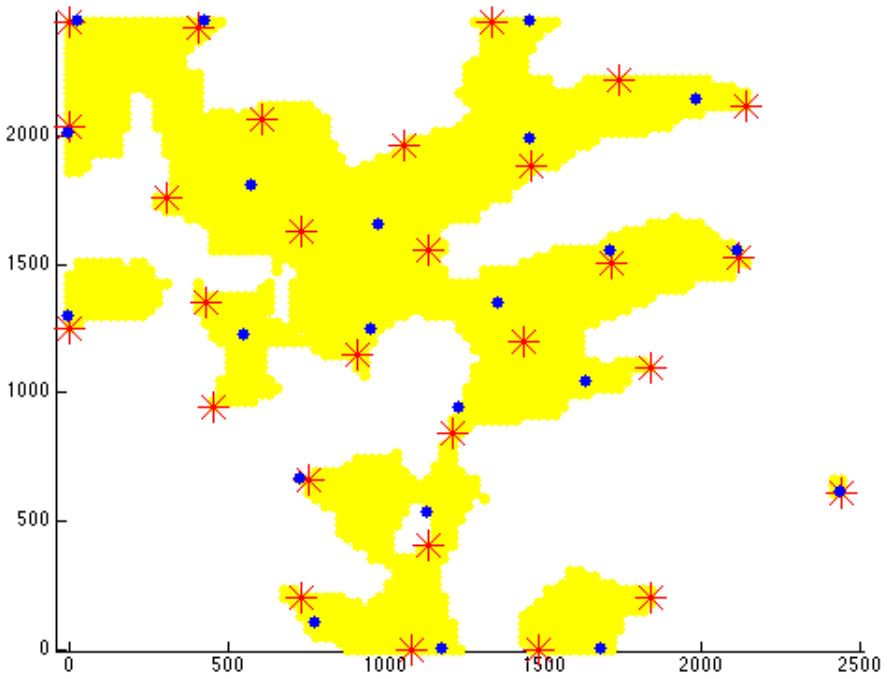


Figure 7.2. Optimized layouts in case of unlimited maximum number of turbines. The yellow area represents the feasible area, the blue dots represent the layout found by *WindPRO's Optimize*, while the red stars are the layouts computed by our tool. *WindPRO's Optimize* is able to locate only 22 turbines vs 28 of our tool.

WindPRO		Initial local-search heuristic			MILP heuristic		
n.turb located	layout profit (MWh/y)	n.turbines located	layout profit (MWh/y)	Improvement wrt WindPRO	n. turb located	layout profit (MWh/y)	Improvement wrt WindPRO
22	200 412	28	231 641	+15.6 %	28	247 569	+23.5%

Table 7.2. Comparison between *Optimize* layout and our layout with unlimited number of turbines.

Our next validation tests use the same settings as before, but the (maximum) number of turbines is now fixed to 5, 10, 15, 20 and 22, respectively. These tests are intended to quantify the benefits of choosing better positions when building the same number of turbines. The following plots compare the layouts found by the two optimizers.

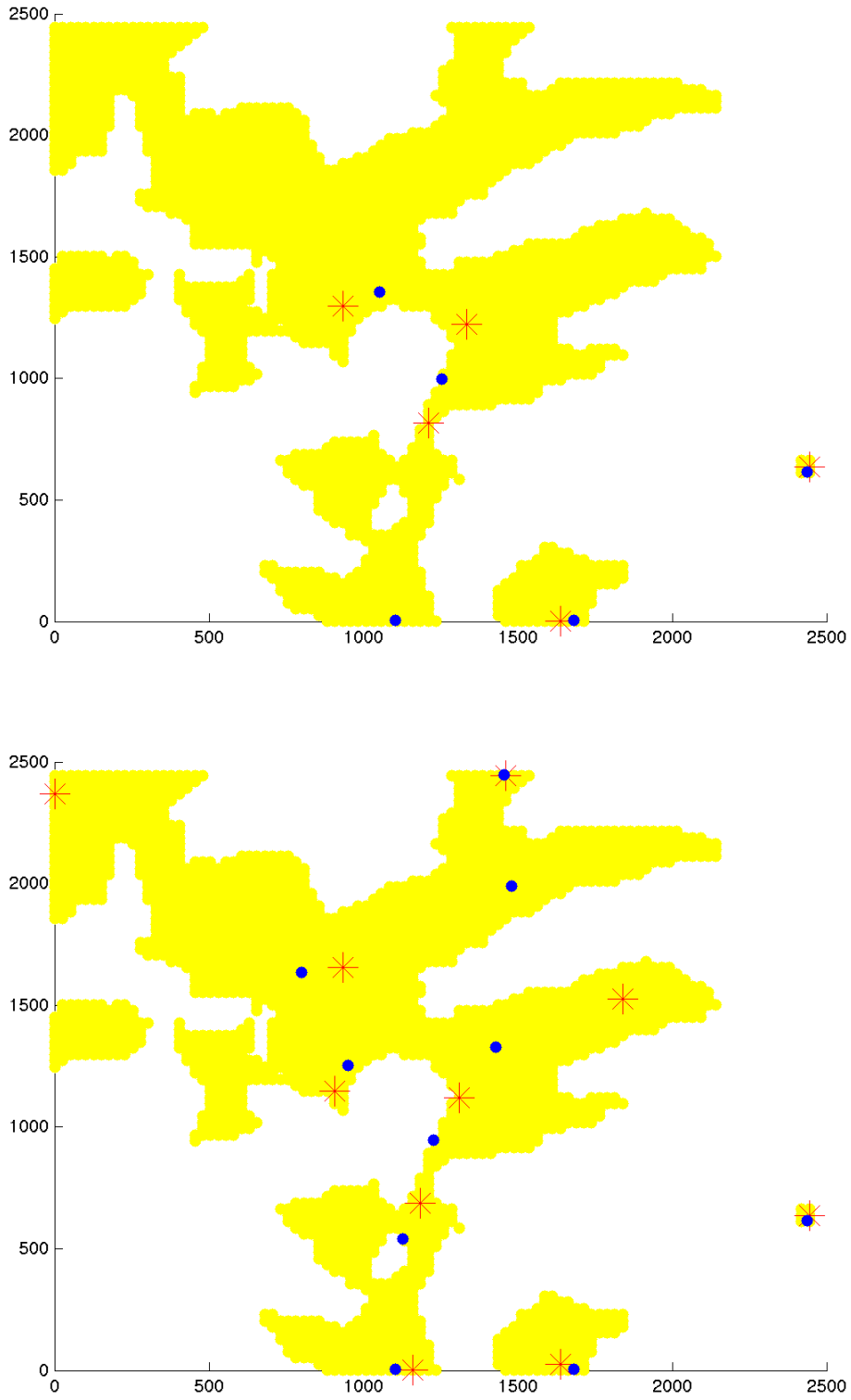


Figure 7.3. Compared layouts in case of maximum number of turbines equal to 5 and 10. The yellow area represents the feasible region, the blue dots represent the layout found by *WindPRO's Optimize*, while the red stars are the layouts computed by our tool.

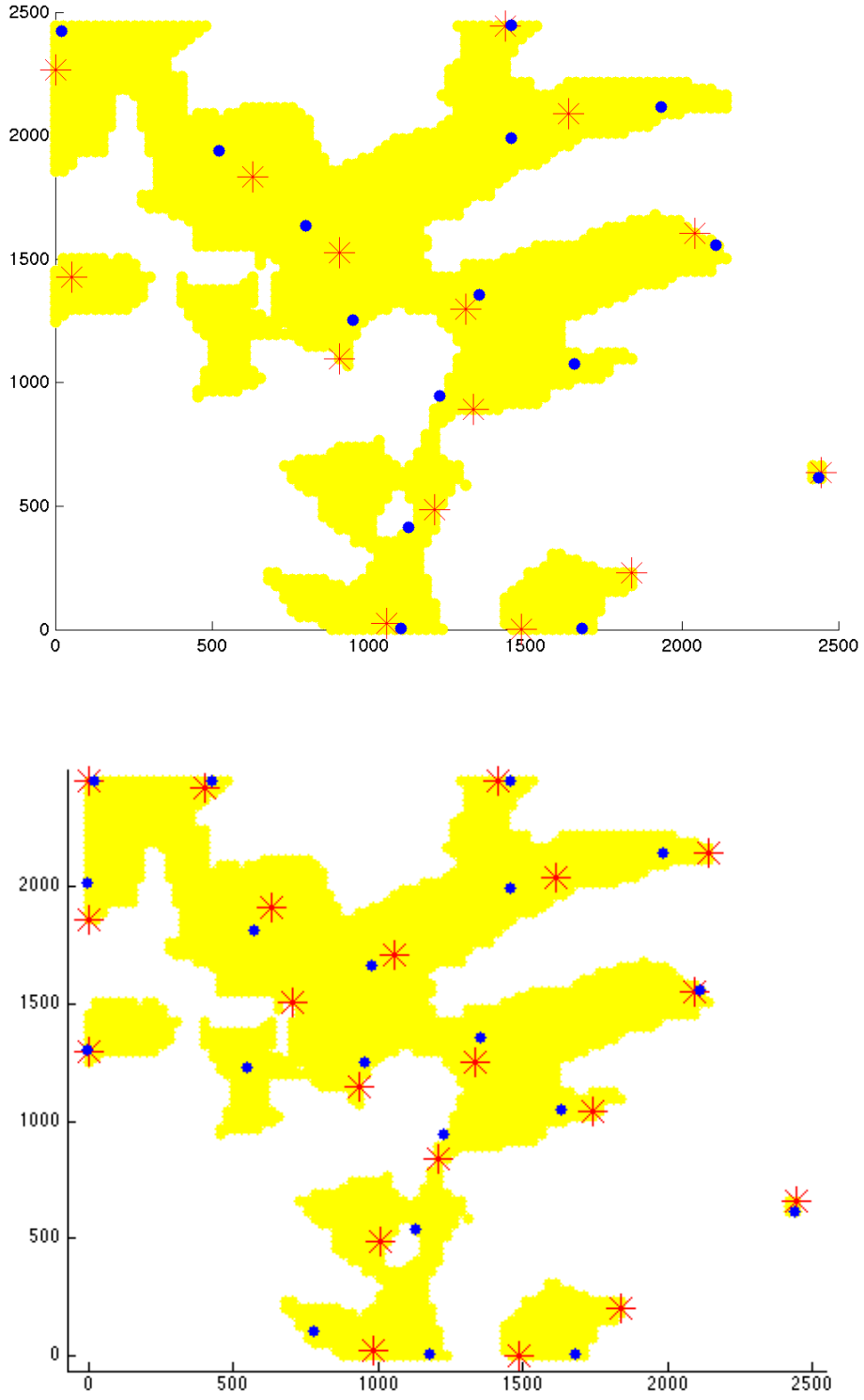


Figure 7.4. Compared layouts in case of maximum number of turbines equal to 15 and 20. The yellow area represents the feasible region, the blue dots represent the layout found by *WindPRO's Optimize*, while the red stars are the layouts computed by our tool.

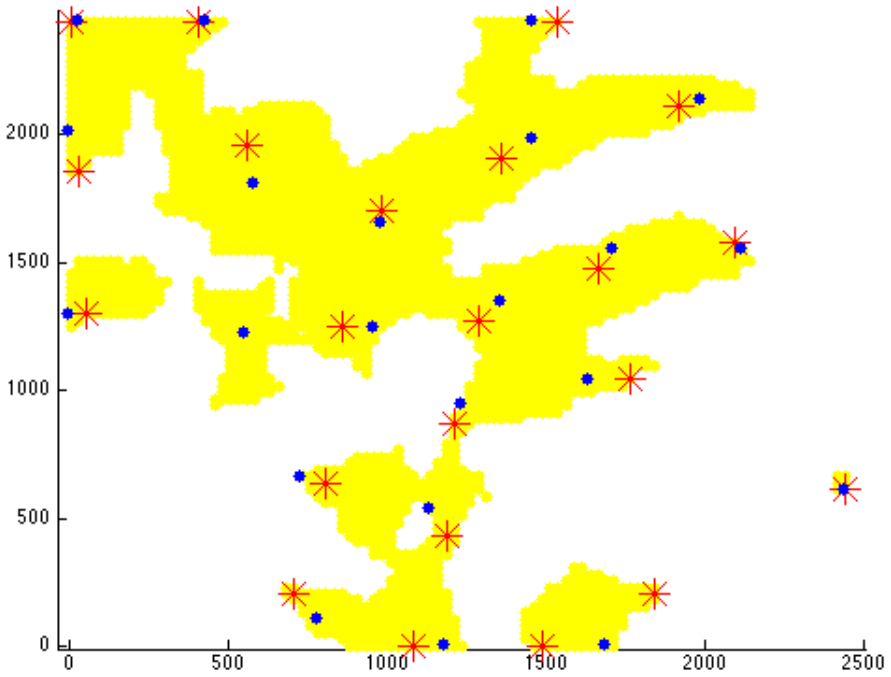


Figure 7.5. Compared layouts in case of maximum number of turbines equal to 22. The yellow area represents the feasible region, the blue dots represent the layout found by *WindPRO's Optimize*, while the red stars are the layouts computed by our tool.

Table 7.3 compares the profits of the layouts found by the two tools. The table shows that our results consistently outperform the one from *WindPRO*. Note that in this context an improvement of $1 - 2\%$ is really interesting since it can be translated in a huge amount of money.

N_{MAX}	WindPRO's layout profit (MWh/y)	MILP heuristic layout profit (MWh/y)	Improvement wrt WindPRO
5	51 776	52 123	+0.7%
10	99 577	101 024	+1.5%
15	144 789	146 454	+1.1%
20	185 252	188 782	+1.9%
22	200 412	205 039	+2.3%

Table 7.3. Comparison between *Optimize* and our full heuristic (based on MILP). Profit is computed in 5 optimized layouts of 5, 10, 15, 20, and 22 turbines. Note that *WindPRO* is not able to put more than 22 turbines in the site.

Another interesting test has been performed by comparing our initial local-search heuristics (only) with the *WindPRO*'s one. In Table 7.4 we compare the *WindPRO's Optimize* layout with the layout given by our 1-opt and 2-opt local-search heuristics (before applying the MILP solver).

N_{MAX}	WindPRO's layout profit (MWh/y)	Initial heuristic layout profit (MWh/y)	Improvement wrt WindPRO
5	51 776	51 983	+0.4%
10	991 577	100 753	+1,2%
15	144 789	146 170	+1,0%
20	185 252	188 424	+1,7%
22	200 412	205 039	+2,3%

Table 7.4. Comparison between *Optimize*'s layout and our initial local search heuristic layout. The profit is computed in 5 test layouts of 5, 10, 15, 20 and 22 turbines.

This last test interesting since the layouts from our heuristics are computed in a matter of seconds, and do not need the use of a MILP solver such as CPLEX. Also in this case, our layouts give a better power production than the commercial software.

7.4 Conclusions

Table 7.5 summarizes the results of our comparison.

WindPRO (WP)				Initial heuristics		MILP heuristic	
max n. turb.	layout profit by WindPRO (MWh/y)	layout profit (our obj value) (MWh/y)	profit difference	layout profit (our obj value) (MWh/y)	Improvement wrt WindPRO	layout profit (our obj value) (MWh/y)	Improvement wrt WindPRO
5	51 695	51 776	0.2%	51 983	0.4 %	52 123	0.7%
10	99 330	99 577	0.2%	100 753	1.2%	101 024	1.5%
15	143 727	144 789	0.7%	146 170	1.0%	146 454	1.1%
20	185 285	185 252	0.0%	188 424	1.7%	188 782	1.9%
22	200 699	200 412	-0.1%	205 039	2.3%	205 039	2.3%
25	200 699	200 412	-0.1%	226 835	13.2%	228 270	13.9%
(unlimited) 28	200 699	200 412	-0.1%	231 641	15.6%	247 569	23.5%

Table 7.5. Summary of the comparison between *Optimize* and our heuristics. Note that *WindPRO's Optimize* is not able to locate more than 22 turbines.

The comparison shows that our optimization tool outperforms *WindPRO* in all tests. This is a remarkable result, as this commercial software is currently used by many wind energy companies all around the world. Our initial heuristics (1-opt and 2-opt) are already able to give, in a matter of seconds, a better result than *windPRO's Optimize*. Refining the solution with our MILP approach results into a further improvement, even if this requires some minutes of computing time (in the worst case).

In a real-world company both the situations we tested can be of interest: sometimes there is a fixed number of turbines to built in a given site, sometimes as many turbines as possible can be built. In the second case, however, some other costs should be taken into account: more turbines produce more energy, but also a bigger initial budget and more maintenance costs are involved. Experts in energy planning and economics are then responsible for the selection of the best strategy (fixed or unlimited number of turbines) for the site of interest. In any case, we stress that even an increase of 1% of the annual energy production is converted in very large money income for the company. Figure 7.6 plots energy prices in Europe (2014).

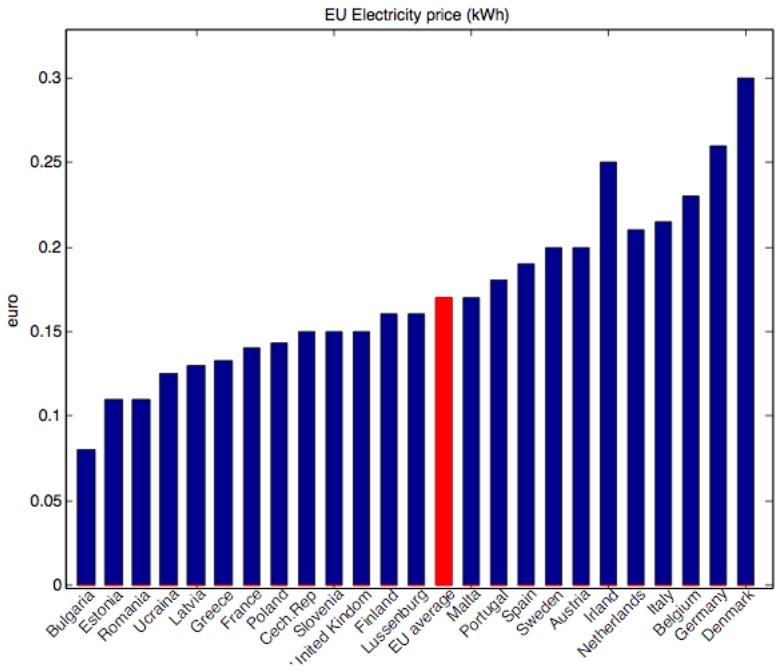


Figure 7.6. European energy prices (€ per kWh) for consumers; data from IAEE Energy Forum, 2014

In Europe the average consumer price is 0.17 €/kWh while operator of wind turbine is typically paying around 0.08€/kWh, hence the price for MWh is 80 €. In the case of maximum number of turbines equal to 20 (Table 7.5), the production with our layout is increasing by 3 530 MWh/y. That means that our incremented production has an economical value of about 280 k€ per year (with the same number of turbines built).

Conclusions and future work 8

In this thesis we studied the wind farm layout problem, a very important optimization problem of great interest in the energy market. We reviewed previous literature on the subject, and in particular the Mixed Integer Linear Programming (MILP) models that were recently proposed. In addition, we proposed a new MILP model that overcomes the main issues of previous proposals and is applicable to the very large scale instances arising in practical applications. Several heuristics have been proposed, implemented in C programming language, and computationally compared on both real-world and synthetic instances. Validation experiments showed that our method compares favorably with a commercial software widely used by planners.

The main issues found during the thesis were as follows. First of all, we needed to describe the wake effect between turbines in both offshore and onshore cases, so we adapted the well-known Jensen's model to a 3D scenario. Secondly, we needed to model our optimization problem in a computationally manageable way. The models for the literature, indeed, turned out to be practically useless already when the number of possible positions was around 100. We then proposed a new model producing weaker LP-relaxation bounds but with a much faster resolution of each branch and bound node, that is able to deal with bigger instances. In addition, we developed some heuristics to deal with even more possible positions (10000+) in a few minutes on a standard PC. Lastly, we compared our tool with a commercial software. This required to get our cost functions comparable to be able to use the same real-world input.

Future work should address various extensions of the basic problem that we studied in the present thesis. Since the optimization of wind farm layout is a huge problem with several additional constraints, in the following we give just a sketch of some possible extensions.

A main issue in the offshore case is the optimized connection among the built turbines through cables. Indeed, cable costs and constraints tend to create a compact layout, while interference tends to spread turbines all over the site. In addition, even with a fixed layout, finding a feasible cable connection of minimum cost is nontrivial. Many cable constraints, indeed, should be taken into account, such as length, capacities, different types of cable, proper connection with the drain, accessibility for maintenance, and so on.

The offshore layout should also be connected to the coast to transfer the produced energy. Another challenging optimization task could be then to find the optimal position of the collecting drain, that should be at the same time as close as possible to the coast, as close as possible to each string of turbines, and reachable by boats.

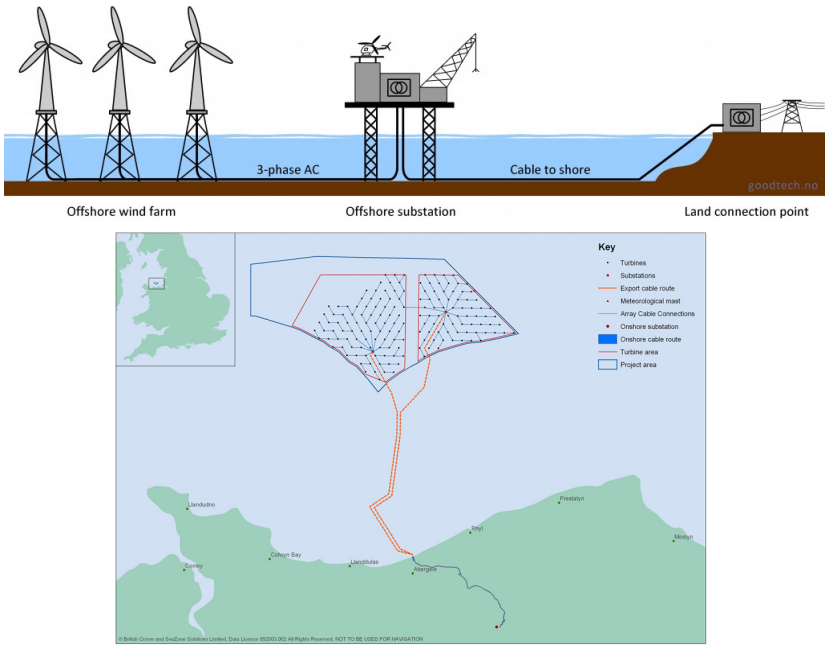


Figure 8.1. Example of a connection among turbines, drain and coast.

Cable price is not the only cost in building a wind farm. That is why, while maximizing the power output, one should also consider minimizing building costs, for example, amount of work and material, building roads and transformer stations, external power grid, and cost of foundations. About foundation, in the onshore case in this thesis we considered that a turbine could not be built where the terrain is too sloping, by excluding the associated positions. It could be of interest in the offshore case, for example, to consider how the optimal layout would change by considering that the foundation cost depends on the sea depth in each position. Since different types of foundation exist, it could be a challenging optimization problem also to optimize the decision of which type of foundation use in each position.

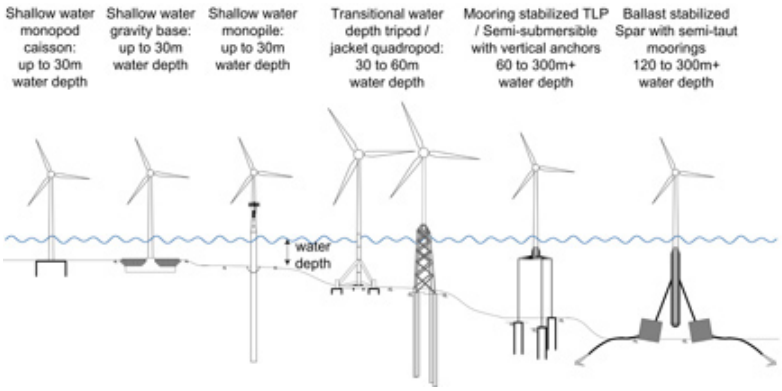


Figure 8.2. Different type of foundations

Different classes of turbines exist, depending on the power they can handle, their dimensions and their costs. It could be of interest, indeed, to optimize the layout considering that different types of turbines can be built in each position.

Finally, in the onshore case in particular, one should also consider the impact that the wind farm would have on the population living nearby. In particular, there are regulations stating maximum sound levels originating from wind turbines at nearby houses or other sensitive areas.

The sound level depends not only on the distance between the wind farm and the sensitive areas, but also the number on turbines built and their dimensions. As a consequence, also sound levels should be taken into account in the optimization.

Bibliography

[1] Vattenfall, *Facts about Horns Rev wind farm*, 2013. [Online]. Available: http://www.vattenfall.dk/da/fakta.htm

[2] Vattenfall, *VATTENFALL*, 2014. [Online]. Available: http://corporate.vattenfall.com/

[3] N. Jensen, "A note on wind generator interaction," Technical Report Risø-M- 2411(EN), Risø National Laboratory, Roskilde, Denmark, Tech. Rep., 1983.

[4] Donovan, "Wind farm optimization," *Proceedings of the 40th Annual ORSNZ Conference*, pp. 196–205, 2005.

[5] R. Archer, G. Nates, S. Donovan, and H. Waterer, "Wind turbine interference in a wind farm layout optimization – mixed integer linear programming model," *Wind Engineering*, vol. 35, no.2, pp. 165–178, 2011.

[6] IBM ILOG CPLEX, *Optimization Studio*, 2013. [Online]. Available: http://www.cplex.com

[7] A. Kusiak and Z. Song., "Design of wind farm layout for maximum wind energy capture." *Renewable Energy*, no. 35, p. 685–694, 2010.

[8] P. H. Madsen, "Introduction to the IEC," Risø DTU National Laboratory for Sustainable Energy, Tech. Rep., 2008. [Online]. Available: http://www.windpower.org/download/461/Introduction_to_the_IEC.pdf

[9] EMD, *WindPRO*, 2014. [Online]. Available: http://www.emd.dk/windpro/frontpage

[10] N. G. Mortensen, "Wind farm AEP and wake loss calculations," Risø DTU, Denmark, Tech. Rep., 2010.

[11] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization.* New York, NY, USA: Wiley-Interscience, 1988.

[12] G. Dantzig, *Linear programming and extensions.* Princeton Univ. Press, 1963.

[13] F. Glover, "Improved linear integer programming formulations of nonlinear integer problems," *Management Science*, vol. 22, pp. 455–460, 1975.

[14] G. A. Croes, "A method for solving traveling salesman problems." *Operations Research*, vol. 6, pp. 791–812, 1958.

[15] F. Glover, "Tabu search: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.

[16] N. Mladenovic and P. Hansen, "Variable neighborhood search," *Computers & OR*, vol. 24, no. 11, pp. 1097–1100, 1997.

[17] M. Fischetti and M. Monaci, "Proximity search for 0-1 mixed-integer convex programming," DEI, University of Padova (submitted), Tech. Rep., 2012.