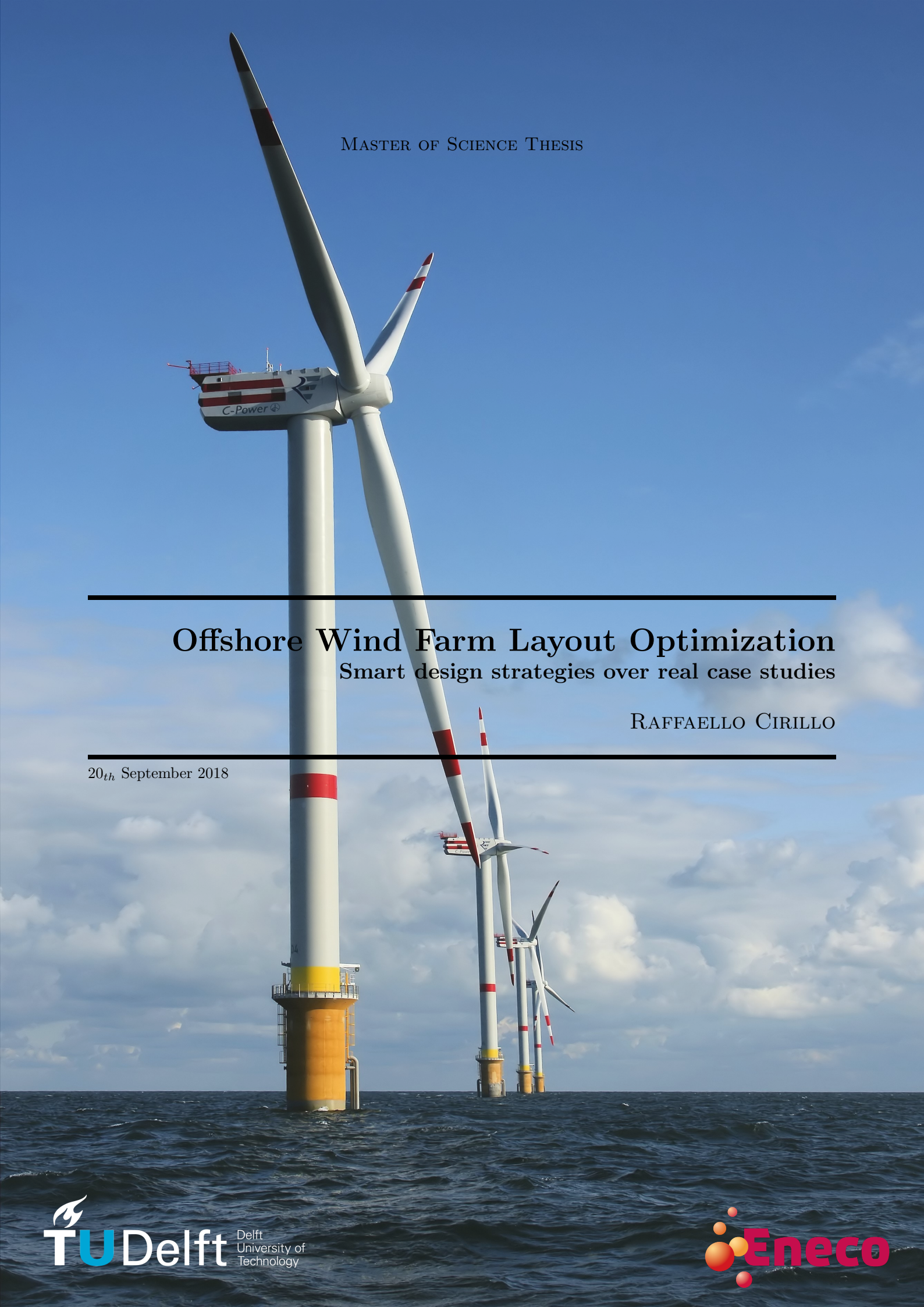


Master of Science Thesis

Offshore Wind Farm Layout Optimization

Smart design strategies over real case studies

Raffaello Cirillo

20_{th} September 2018

TU Delft Delft University of Technology

Eneco

Offshore Wind Farm Layout Optimization

Smart design strategies over real case studies

Master of Science Thesis

For obtaining the degree of Master of Science in Sustainable Energy Technology at Delft University of Technology

Raffaello Cirillo

Student number: 4623770

20_{th} September 2018

Thesis Committee:	Prof. Dr. Simon Watson (chairman)	*TU Delft*
	Dr. Ir. Michiel Zaaijer (supervisor)	*TU Delft*
	Dr. Ir. Sander Hartjes	*TU Delft*
	Ir. Wybren de Vries (company supervisor)	*Eneco B.V. Wind*

Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology

TU Delft Delft University of Technology

Eneco

To my beloved grandfathers Alberto and Raffaello

Summary

Nowadays an increasing number of offshore wind farms (OWF) is getting built. Clear advantages w.r.t. onshore farms, such as higher energy yields, less visual pollution, less restrictions about operational noise and relatively low environmental impact all contribute to the spread of this technology. Several disciplines are considered into this subject. The design of the support structure requires civil engineering knowledge and the aerodynamic design of a blade is connected to the concept of an airfoil. Moreover, electrical and control engineering are involved, as well as economics.
The manual and sequential design approach used so far is not sufficient to guarantee optimized systems because interactions between the different components are disregarded. Therefore, engineering companies are looking for a multidisciplinary, modular, user-friendly and easy tool to be applied during the preliminary design assessment. This should take into consideration almost every crucial aspect of OWF design. Moreover, it should be *able to perform some overall layout optimization*, i.e. providing the best turbine positioning w.r.t. the energy yield, the costs, the cables, the installation and the O&M, ensuring some good constraint-handling techniques and guaranteeing a sufficient degree of robustness, precision, flexibility and speed.
Therefore, the concept of MDAO (Multisciplinary Design Analysis and Optimization) workflow has been introduced. With respect to the common sequential design procedure, which optimizes the wind farm components *separately*, the MDAO analysis helps the user deal with the *interactions* between different design areas whilst *automating* the design process. In other words, this approach couples the modules which analyze the different wind farm design areas - aerodynamics, support structures, cable topology etc. - with the aim of evaluating the cost and the overall performance *of the whole system*. The result is a design where all the parts are jointly optimized.
Within the MDAO domain, two components are identified. The former is called *analysis block* and it comprehends all the modular tools which refer to a specific physical/economic discipline; the latter is the *driver*, i.e. an optimizing procedure which calls the analysis block in each iteration [9].

This thesis focuses mostly on the analysis of the *driver*. In particular, the first goal is to identify the *best optimizing strategy* to be adopted, thanks to several quantitative assessment criteria. The driver is made of three parts: the *initialization* of the optimization (i.e. the initial guess), the way constraints are implemented (*constraint-handling techniques* - CHT) and dealt with and the choice of the *algorithm*. The objective which is considered is the minimization of the LCOE, while the constraints are some minimum distance between the turbines ($4D_{rotor}$) and all of them being within the OWF boundaries. Two real case studies are considered, i.e. Eneco Prinses Amalia Wind Park (PAWP) and Eneco Luchterduinen (EL). No clear guidelines exist to give the designer any clue about *what optimizing strategy performs best when dealing with OWFLO*. Therefore, several combinations have been tried. Six different initializations have been coupled with three algorithms (Genetic, Particle Swarm and Differential Evolutionary) and four CHTs, i.e. static, dynamic, adaptive penalty functions and a repair mechanism. Among all the analyzed combinations, the best performances were achieved by a Differential Evolutionary Algorithm (DEA) and a Genetic Algorithm (GA) coupled with a *feasible initialization* and a *repair mechanism* as constraint-handling technique.
After having identified the best procedure to implement into the driver, the influence of the wind direction sampling stepsize is investigated. It has been found that when the number of sampling sectors is above 24 (corresponding to $\theta = 15°$) no further precision of the wake deficits is achieved. In other words, when a sufficiently high angle discretization is used, the results from the optimization are more trustful because the turbines belonging to a wake are always *detected*.

The second research goal is studying to what extent the different design areas - aerodynamics, support structure, cable topology etc. - affect the result of the optimization. The analysis shows that no general answer can be given about what discipline mostly affects the LCOE calculation, as the results are case-study dependent. It is impossible to say *a priori* which discipline will bias the result the most. However, this provides further evidence of the strength of a multidisciplinary procedure: the MDAO workflow always tries to reach a trade-off between sometimes conflicting goals, which are dependent on the case study under consideration.

The third research objective is to see whether the chosen optimizing procedures are able to deal with more complex design situations, such as more constraints or longer design vectors. Three challenges, characterized by an increasing difficulty, have been considered over the new case study of Borssele III. The first one is similar to the previous two case studies of EL and PAWP, i.e. same constraints, objective and design vector; the second one takes into account the presence of a forbidden zone *inside* the boundaries due to the presence of cable routes; the third one adds the number of turbines to the design vector, to see whether the optimizer is capable of finding the optimal number of turbines as well as their optimal layout. All the three challenges were successful and highlighted the ability of the driver to deal with more complex situation. This is promising in case the designer is called to face additional constraints, which are likely to occurr in real projects.

In the end, an additional analysis on the effect of neighbouring wind farms on the layout optimization of Borssele has been done. According to the Jensen model and by adopting the same optimizing strategies as in the three challenges, it was observed that considering the presence of other plants affects the AEP. A comparison between the optimized layout in *undisturbed conditions* and the optimized layout in *disturbed conditions* highlighted that in case during the design process of an OWF it is known that there are (or will be) neighbouring wind farms, this should be taken into account *within* the optimizer, as the two optimal values are too different from each other.

Acknowledgements

This two year journey has eventually come to an end. This experience gave me much more than I could ever expect. I grew up both from the personal and the professional point of view: I met a lot of fantastic people who introduced me to the lifestyle of other countries; I learned many things on the topic I care the most, i.e. sustainable energy; I worked for two prestigious Dutch companies and further realised that, despite this period of crisis across Europe, no barriers against foreigners should be imposed. That is why my utmost gratitude firstly goes to the country of the Netherlands, for welcoming me and never letting me feel a stranger.

I put much effort into this research. For the very first time in my life, I really enjoyed getting up early in the morning and going to work. I loved the topic I had to deal with and I had a very good time at Eneco. It is with true pleasure and sincere gratitude that I thank my company supervisor, Ir. Wybren de Vries, for allowing me to carry out this interesting thesis. His help, support and empathy really motivated me to do my best. I would also like to thank Dr. Ir. René Bos, for his precious pieces of advice about optimizing algorithms and coding issues, together with his cheerfulness. I also got in contact with some colleagues I spent many fun moments with, in particular Ir. Adam Morón (thank you for your sense of humour and suggestions), Nicole Krolis and Ir. Bas van den Kieboom. Working on the thesis was less hard than I thought!

I will never thank enough my daily supervisor, Dr. Ir. Michiel Zaaijer. His availability, patience, critical eye and commitment really pushed me to go more in depth, accurately investigate on my results and question them. I would also like to express my gratitude to Ph.D. candidate Sebastian Sanchez Perez-Moreno from TU Delft. Thank you for always being available every time I had some issues on the codes or some doubts. Without you, everything would have been (much) more difficult.

On a more personal note, I would like to thank my (old) housemate, Livia, for bearing me during our year together. I am sure it was not a very easy task! I also thank my fellow graduate students, for the time we spent laughing and sustaining each other, and my friends from Italy.

My sincere gratitude goes to my cousin Elisa. Your sense of humour, help and absurd lifestyle were really good for my mood! In addition, thank you mum and dad, for always being next to me despite the distance and making me feel at home every day.
Last but not least, I would like to thank my sister Maria (we do not need words) and my beloved girlfriend, Silvia, for her support, patience and true love which still accompanies me every day of my life.

Dank je wel! Grazie!

Delft, 20_{th} September 2018

Raffaello

Contents

List of Figures

List of Tables

Nomenclature

List of abbreviations

Symbol	Description
AEP	Annual Energy Production
CDF	Cumulative Distribution Function
CFD	Computational Fluid Dynamics
CHT	Constraint-Handling Techniques
COP	Cost Of Power
CR	Crossover Rate
DEA	Differential Evolutionary Algorithm
e.g.	exempli gratia
EL	Eneco Luchterduinen
GA	Genetic Algorithm
HAT	Highest Astronomical Tide
i.e.	id est
KBE	Knowledge Based Engineering
LAT	Lowest Astronomical Tide
LCOE	Levelised Cost of Energy
MCA	Multi Criteria Analysis
MDAO	Multidisciplinary Design Analysis and Optimization
O&M	Operation and Maintenance
OT	Offshore Transformer
OWF	Offshore Wind Farm
OWFICTP	Offshore Wind Farm Infield Cable Topology Problem
OWFLO	Offshore Wind Farm Layout Optimization
PAWP	Prinses Amalia WindPark
PC	Personal Computer
PSO	Particle Swarm Optimization
RNA	Rotor Nacelle Assembly
SAW	Simple Additive Weighting
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
TP	Transition Piece
WACC	Weighted Average Cost of Capital
WF	Wind Farm
WINDOW	Wind farm INtegrated Design and Optimization Workflow
w.r.t.	with respect to

List of mathematical symbols

Symbol	Description	Units
a	annuity factor	-
A	Alternatives in MCA	
$A_{nacelle,front}$	front area nacelle	m^2
$C_{1,2,3}$	learning factor	-
C_{dec}	decommissioning costs	€
C_{inv}	investment costs	€
$C_{O\&M}$	O&M costs	€
$C(T)$	penalty function factor	-
c_T	thrust coefficient	-
D	diameter	$[m]$
d_{50}	particle size distribution	$[m]$
E_y	annual energy yield	$[GWh]$
f	fitness function	-
F	scaling factor	-
$g(\mathbf{x})$	inequality constraint function	-
g_{best}	best PSO particle global position	-
$h(\mathbf{x})$	equality constraint function	-
$H_{s_{1-year}}$	1 year max significant wave height	$[m]$
$H_{s_{50-year}}$	50 year max significant wave height	$[m]$
I_a	ambient turbulence intensity	-
k	wake decay coefficient	$[m^{-1}]$
K	Forbidden zone offshore transformer	-
$L_{harbour}$	distance from OWF to harbour	$[km]$
L_{min}	minimum distance constraint	m
$L_{onshore}$	onshore transmission distance	$[km]$
L_{total}	Total distance from OWF to grid	$[km]$
M	Assessment criterion	
m_{RNA}	rotor-nacelle-assembly mass	$[kg]$
n_{spots}	number of spots in the random-triangled grid	-
$n_{substations}$	number of substations	-
N_T	number of turbines	-
p_{best}	best PSO particle individual position	-
P_{rated}	rated power	$[MW]$
P_{size}	population size	-
r	real interest rate	-
r_{ij}	score	-
R_{rotor}	rotor radius	$[m]$
R_i	benefit attributes for MCA analysis	-
$\Re$	real numbers set	-
S	wind farm boundaries	-
T	economic lifetime of a project	$[years]$
T	iteration number	-
T_{rotor}	rotor thrust	$[N]$
U	wind speed	$[m.s^{-1}]$
U_0	undisturbed wind speed	$[m.s^{-1}]$
$U_{current}$	Sea current speed	$[m.s^{-1}]$
U_{cut-in}	cut-in wind speed	$[m.s^{-1}]$
$U_{cut-out}$	cut-out wind speed	$[m.s^{-1}]$
u_G	DEA trial vector	-
U_{rated}	rated wind speed	$[m.s^{-1}]$
v	speed for PSO	-
v_G	DEA donor vector	-
$V_{generator}$	generator voltage	$[V]$
$V_{infield}$	infield rated voltage	$[V]$
$V_{transmission}$	transmission voltage	$[V]$
$\mathbf{x}$	design vector	-

x	x-coordinate	$[m]$
x^t	PSO particle position at iter. t	-
$\mathbf{x}^*$	optimum design vector	-
y	y-coordinate	$[m]$
z_0	surface roughness	m
z_{hub}	hub height	$[m]$
Z	Forbidden zone due to cable routes	-

List of greek symbols

Symbol	**Description**	**Units**
α	dynamic/adaptive penalty exponent	
$\delta\alpha_{partial}$	wake incidence angle	$[^\circ]$
γ	safety factor	-
θ	wind direction sampling stepsize	$[^\circ]$
λ	adaptive penalty constant	-
μ	mean	-
ν	inflation rate	-
σ	standard deviation	[m]
σ_{cr}	material yield stress	[MPa]
σ_{factor}	stress factor	-
σ_{MAX}	maximum occurring stress	[MPa]
ϕ	friction angle	$[^\circ]$
ω	weight in MCA	-

Chapter 1

Introduction

1.1 Overview

At the end of 2016, the total installed offshore wind capacity was 14.38 GW, being present in 14 countries around the world [4]. The biggest contributions come from, in descending order, UK (36%), Germany (29%), China (11%), Denmark (8.8%), the Netherlands (7.8%), Belgium (5%) and Sweden (1.4%) [4]. Offshore wind energy is convincingly showing off as one of the most promising technologies driving the transition towards a more sustainable future.
Nowadays an increasing number of offshore wind farms (OWF) is getting built. Clear advantages w.r.t. onshore farms, such as higher energy yields, less visual pollution, less restrictions about operational noise and relatively low environmental impact all contribute to the spread of this technology. Several disciplines are considered into this subject. The design of the support structure requires civil engineering knowledge and the aerodynamic design of a blade is connected to the concept of an airfoil. Moreover, electrical and control engineering are involved, as well as economics.
The manual and sequential design approach used so far is not sufficient to guarantee optimized systems because interactions between the different components are disregarded [5]. Therefore, a *multidisciplinary* approach is required. Companies are very interested in developing a smart design procedure for OWF. This implies the need for some overall optimization: a trade-off between the desired energy output and the costs is the key for further development in the wind energy sector [6].

1.2 Problem analysis

Due to the high complexity of designing an OWF, the standard practice is to follow a *decoupled* and *sequential* design procedure [5]. Indeed, there are several commercial softwares dedicated to the design of OWFs. These are, for example, WindPRO or TopFarm [5]; nonetheless, none of them is based on a multi-objective algorithm which is able to optimize sometimes conflicting goals. For example, larger turbine spacing ensures higher energy production but increases the cabling cost; a layout which has been optimized by the aerodynamic point of view may not consider the bathymetry and the related structural issues. However, sophisticated approaches (such as computational fluid-dynamics - CFD) have high computational costs and carrying out a multidisciplinary design with such approaches would be infeasible.
Instead, a better integrated methodology would yield a consistent reduction of the cost of energy, strengthening the competitiveness of offshore wind in the future world energy matrix [5]. Therefore, engineering companies are looking for a multidisciplinary, modular, user-friendly and easy tool to be applied during the preliminary design assessment. This should take into consideration almost every crucial aspect of OWF design. Moreover, it should be *able to perform some overall layout optimization*, i.e. providing the best turbine positioning w.r.t. the energy yield, the costs, the cables, the installation and the O&M, ensuring some good constraint-handling techniques and guaranteeing a sufficient degree of robustness, precision, flexibility and speed.

Several authors investigated on the OWF layout optimization problem (OWFLO). A pioneering paper was published in 1992 by Mosetti et al. [7], who tried to maximize the annual energy production of a wind farm by optimizing the position of the wind turbines by the means of a *genetic algorithm*. From there onwards, a

large number of articles was released. Several algorithms have been explored, as well as different constraint-handling techniques (CHT), addressing the layout optimization problem. However, only a few authors approach this problem from a *multidisciplinary* perspective. Moreover, no clear guidelines exist to give the designer any solid clues about what optimizing strategy performs best when dealing with OWFLO. There are in fact a large number of available *combinations* in terms of: initialization of the optimization (i.e. the initial guess), the way constraints are implemented and dealt with and the choice of the algorithm.

Secondly, a line of research which has not been fully explored is studying to what extent the different design areas - aerodynamics, support structure, cable topology etc. - affect the result of the optimization. As shown in [8], switching off one design aspect instead of another significantly affects the optimized result. When the layout optimization problem is analyzed by a multidisciplinary perspective, it is then crucial to try to quantify *how* the optimization solves the trade-offs among competing disciplines [1]. One good example can be found in [1], where the author performs the optimization of an OWF layout w.r.t. *different objectives*, i.e. the maximization of the energy yield, the minimization of the cost of the support structures and the cost of the infield cables. However, that analysis is not properly meant to assess which sector contributes the most to a change in the layout, but it is carried out to make a comparison between a *sequential design procedure* and a multidisciplinary optimization. On the other hand, *quantifying the effect* of the involved engineering disciplines on the optimization would help the designer define what design areas mostly bias the final result. Therefore, it would be desirable to illustrate some guidelines on which disciplines need special attention on a preliminary design stage.

1.3 Objective

The objective of this research can be summarized by the following research questions:

1. *What is the best optimizing routine - in terms of initialization, CHT and choice of the algorithm - to be coupled with a multidisciplinary design tool which can be satisfactorily applied to the future design procedure of engineering companies?*

   The goal is to illustrate the advantages and disadvantages of a large number of optimizing combinations, in order to wisely select the best one.

2. *Which design areas - aerodynamic wake models, support structure design and cable topology - mostly affect the optimization and to what extent?*

   This question aims to quantify the effect that the design disciplines have on the overall layout optimization.

3. The third research question had not been included in the problem analysis, but it has been added afterwards. This is as follows:
   is the selected optimizing procedure from Question 1 able to deal with very complex design requirements?

   After the best design methodology has been identified, this analysis takes place to *challenge* the selected strategy to work under more difficult situations, such as a higher number of constraints or different variables. The goal is demonstrating the flexibility of the chosen technique when dealing with design requirements which are closer to reality.

1.4 Methodology

The Wind Energy Group at TU Delft is in possession of a Python modular tool, called WINDOW (i.e. " *Wind farm INtegrated Design and Optimization Workflow*"), developed by Ph.D. candidate S.S. Perez-Moreno [9], which takes into consideration all the design areas listed above. This includes a cost model, several aerodynamic wake models, a support structure package, an O&M (simplified) model and a cable topology built-in optimizer. This tool is the starting point to perform the optimization. In other words, *all* the listed disciplines contribute to carry out a *multidisciplinary optimization*. The WINDOW tool in fact belongs to the so called multi-disciplinary design analysis and optimization workflow (MDAO workflow [9]). An MDAO workflow couples WINDOW - which provides the *physical* and *economic* description of the OWF - to the optimizer, which then finds the best layout based on a trade-off between conflicting design areas. The theoretical background of WINDOW is available in [6], [9] and [8].
The methodology is as follows. The validation of WINDOW had already been performed in [8] over the OWF of Horns Rev. Although some more in-depth research on how to improve the different packages would be desiderable, no significant corrections are expected to take place in the code. However, the first step is at least

checking whether the order of magnitude of the results from WINDOW are in compliance with some real cases. The chosen case studies are Eneco's OWFs Prinses Amalia Wind Park (PAWP) and Eneco Luchterduinen (EL). After this preliminary validation, the analysis can shift towards the optimization.
The second step consists of selecting the optimizing *techniques* to be implemented in the MDAO workflow. This is done by taking care of the general OWFLO strategies which are currently widely adopted in literature. The goodness of the coded algorithms is then tested over some known functions.
Thirdly, different ways to start the optimization are investigated and implemented in Python, as well as the constraint-handling techniques (CHT). At this stage, a large number of combinations of "initialization-CHT-algorithm" is tested and assessed by some criteria, defined by the author. The goal is to carry out a multi-criteria analysis (MCA) to compare [9] these combinations and find the best one.
Once that the best strategy is selected, the effect of the different design areas on the overall result is investigated. In the end, a new, more complex case study (i.e. an OWF currently being designed but not built) is tested, to challenge the chosen procedure to work under more difficult situations.

1.5 Thesis outline

The layout of this work is as follows:

- Chapter 1 consists of the introduction, the problem statement, the research question and the methodology;
- Chapter 2 presents what is the offshore wind farm layout optimization problem. In particular, it firstly gives a quick overview of the theory of optimization to facilitate the comprehension of the OWFLO;
- Chapter 3 provides an overview of the MDAO workflow and how this is related both to the WINDOW tool and the optimizer. On a later stage, the case studies, i.e. Prinses Amalia Wind Park and Eneco Luchterduinen are presented, as well as the validation of WINDOW;
- Chapter 4 explains which algorithms have been selected and the reason under the choice. On a later stage, some tests over known functions are performed to verify the correct operation of the coded optimizations;
- Chapter 5 is the core of the work as it deals with the optimization setup. A detailed explanation on the different ways to start the optimization and to handle constraints, as well as an in-depth description on how the algorithms were applied to the OWFLO problem, is provided;
- Chapter 6 shows the results. The best optimizing strategy is selected by means of some ranking criteria. Moreover, the influence that both the wind direction sampling and the design areas have on the final results is investigated;
- Chapter 7 analyses how the chosen optimizing strategy behaves on a *new and more complicated use case*;
- Chapter 8 draws the conclusions and future recommendations.

Chapter 2

The Offshore Wind Farm Layout Optimization problem (OWFLO)

2.1 Introduction to the chapter

This chapter presents an outline of the offshore wind farm layout optimization problem (OWFLO) and is divided into two parts: the first section presents the definition of *optimization* and the related terminology; the second section shows the main ingredients to perform OWFLO, i.e. the objective function, the design vector and the constraints.

2.2 Optimization: definition and terminology

To enhance the comprehension of this work, a quick overview about *design optimization* is given. A certain design problem is described by a set of *design variables* $x_1, x_2, ..., x_n$ gathered in the *design vector* $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ belonging to the n dimensional space $\mathbf{X} \subseteq \Re^{\mathbf{n}}$ (design space) [10]. Given a certain design problem, its solution is a *combination* of the design variables. Among all the possible combinations, the solution which provides the best achievable performance is the *optimized solution*. The performance of a set of design variables is assessed by a function f called *objective function*. Optimizing means finding either the highest (i.e. *maximization*) or the lowest (i.e. *minimization*) value of the objective function in order to obtain the best design vector $\mathbf{x}$. In this work, minimization is considered. The mathematical formulation of an optimization problem is as follows [10]:

An optimization problem consists of finding an $\mathbf{x}^* \in X$ *such that, for* $f : X \Rightarrow \Re, f(\mathbf{x}^*) = min(f(\mathbf{x}))$ *for* $\mathbf{x} \in X$. [10]

In real-life problems, the design variables are often subjected to some limitations in the design space. These limitations are called *constraints* and are represented by the following functional relations: $h(\mathbf{x}) = 0$ and $g(\mathbf{x}) \leq 0$. The former is the common way to represent *equality constraints*, while the latter symbolizes *inequality constraints* [10]. An optimization problem is defined as *constrained* if it is subjected to those limitations, otherwise it is called *unconstrained*. A solution which does not violate the constraints is named *feasible*. To summarize:

$$\begin{aligned} find \quad & x^* : f(\mathbf{x}^*) = min(f(\mathbf{x})) \\ subjected \quad to \quad & h(\mathbf{x}) = 0, \quad g(\mathbf{x}) \leq 0 \\ with \quad & \mathbf{x} = [x_1, x_2, ..., x_n]^T \end{aligned} \tag{2.1}$$

The definition above is useful as it states that *an optimization problem is described by three ingredients: the design vector, the objective function and the constraints.*
It is important to point out that some complex design problems require the minimization (or maximization) of *more* than one objective function. If such a situation takes place, then it is called *multi-objective* optimization [10]. Due to the often conflicting nature of multiple objectives, the optimal solution is often the *Pareto* front of all the alternatives [9].

2.3 Presentation of OWFLO

Optimizing the layout of offshore wind farms means finding the optimal placement of the turbines w.r.t. the disciplines which contribute to the design. In other words, the optimization consists of moving the turbines inside the wind farm area until a good trade-off between the design areas is found. From the theoretical overview in Paragraph 2.2, the three ingredients which are needed in an optimization problem are the *objective function*, the *design vector* and, if present, the *constraints*.

2.3.1 The objective function

The objective function evaluates the goodness of a certain design. As stated in [6], writing a good objective function involves the employment of *design performance indicators*. These should be able to comprehend both negative properties of the solution, e.g. the costs, and the positive ones, e.g. the generated energy. An essential requirement for the objective function is in fact *combining good and bad properties such as they are correctly weighted and their relative importance is expressed* [6]. One of the most common functions used in literature is the *levelized cost of energy* (LCOE). The formulation of this function is useful as it allows the OWFLO to be treated as a *single objective optimization*, despite the large number of conflicting goals which may affect the final outcome.
The levelised cost of energy is defined as *the cost to breakeven the overall cost of the project and the total revenues, both being actualized for every year of the project lifetime* [11]. This can be expressed as:

$$LCOE = \frac{C_{inv}}{aE_y} + \frac{C_{O\&M}}{E_y} + \frac{C_{dec}(1+r)^{-T}}{aE_y} \qquad [c€.kWh^{-1}] \tag{2.2}$$

where E_y is the annual energy yield, a is the *annuity factor*, T is the lifetime of the project, r is the *real interest rate*; C_{inv} represents the investment costs, $C_{O\&M}$ are the O&M costs, whereas C_{dec} stands for the decommissioning costs. In other words, the LCOE is the ratio of the actualized costs and the energy yield multiplied by a factor which takes into account the real interest rate. The annuity factor is a parameter which allows to compute the present value of a fixed annuity [11]. Equation 2.3 shows the expression which holds for a.

$$a = \sum_{t=1}^{T}(1+r)^{-t} = \frac{1}{r}[1 - (\frac{1}{1+r})^{T}] \qquad [-] \tag{2.3}$$

Several assumptions are made to write Equation 2.2:

- the investment costs do not have to be actualized as all of them are assumed to be entirely paid in year 0;
- the annual energy yield and $C_{O\&M}$ are constant;
- the second element of the sum does not contain the annuity factor because the LCOE describes the cost/revenue balance for every year. The O&M costs/energy yield balance is done every year without the need to actualize;
- the decommissioning costs are supposed to take place only in the year of the shut-down of the farm.

From this short description, it can be concluded that the definition of LCOE attempts to combine all the elements to deal with in the wind farm design.
Although other objectives might have been considered for this thesis, among others the Cost Of Power (COP) and the Annual Energy Production (AEP) [5], the LCOE has been chosen. This is because it is generally acknowledged that a *common goal* (for suppliers and users) is reducing the cost per unit of electricity [6] [12].

2.3.2 The design vector

The design vector for OWFLO varies among the authors and depends on the needs of the designer. For instance, one might be interested in optimizing the layout of a *varying* number of turbines; some authors have investigated on the possibility of having OWFs with different turbine heights [13]. In this work, the number of turbines is fixed and the turbine's specifications are part of the input parameters. However, at the time of writing this thesis, the WINDOW tool is being extended with a built-in Rotor-Nacelle-Assembly (RNA) optimizer.
The considered design vector consists of the $[x, y]$ coordinates of the set number of turbines, N_T. The design vector described for the first time in Paragraph 2.2 then *is* the *layout*:

$$\mathbf{x} = [x_1, x_2, ..., x_n]^T \rightarrow \mathbf{x} = layout = \begin{bmatrix} (x_1, y_1) \\ (x_2, y_3) \\ \vdots \\ (x_{N_T}, y_{N_T}) \end{bmatrix} \tag{2.4}$$

Under this notation, every design variable in the design vector corresponds to a *couple* of points, x and y. *Therefore, the dimension of the problem is* $\Re^{2 \cdot N_{turbines}}$.

2.3.3 The constraints

The constraints which have been considered in this work are an *inequality* and an *equality* constraint. The former is by law a minimum spacing of four times the diameter among the turbines [14]; the latter consists of all the turbines being within the defined wind farm boundaries. The mathematical formulation of constraint 1 is:

$$g_1(\mathbf{x}) : \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} - L_{min} \geq 0, \quad \forall i \neq k \quad i,k \in [0,1,...,N_T] \tag{2.5}$$

the second constraint requires a point (i.e. the turbine) to be inside a convex polygon (i.e. the farm boundaries). Its mathematical formulation is a little bit more complex. Being $[A_1, A_2, ..., A_{N_{vertices}}]$ the vertices of the polygon, point P is placed inside if [15]:

$$h_1(\mathbf{x}) : \sum_{j=1}^{N_{vertices}} = \angle A_j P A_{j+1} = 360° \quad inner \quad point \tag{2.6}$$

or:

$$h_1(\mathbf{x}) : \exists! \quad a_j x^P + b_j y^P + c_j = 0 \quad j = [1, N_{vertices}] \quad point \quad on \quad the \quad edge \tag{2.7}$$

Equation 2.6 states that the sum of all the angles between subsequent points A_j, A_{j+1} whose vertix is P must be 360°; Equation 2.7 states that if the point is on the border, then it has to belong to one of the straight lines which draw the polygon. An easier formulation which enhances the comprehension of the second constraint is:

$$h_1(\mathbf{x}) = (x_i, y_i) \in S \quad \forall i \in [0,1,...,N_T] \tag{2.8}$$

being S the matematical set representing the OWF boundaries. L_{min} is equal to $8R_{rotor}$, i.e. $4D_{rotor}$ [14]. From here onwards, g_1 and g_2 will be called *spacing constraint* and *boundary constraint* respectively. Figure 2.1 provides a visualization.

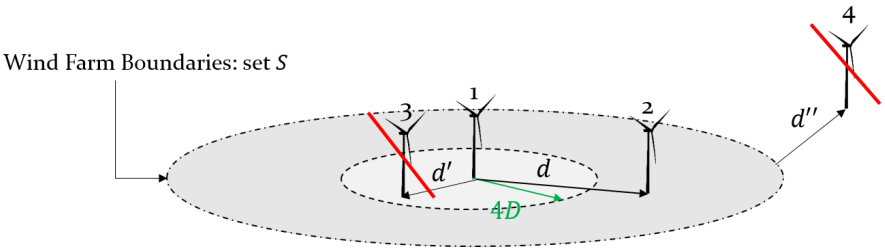


Figure 2.1: *Visualization of the constraints used in this thesis: turbine* 3 *violates the spacing constraint w.r.t. turbine* 1*; turbine* 4 *violates the boundary constraint; turbine* 2 *is in a good position w.r.t. turbine* 1

However, the possibility of handling more constraints *must* be taken into consideration. These might vary from avoiding certain water depths to being far enough from the export cable and the offshore transformer (OT). In this last case, standards require a minimum distance of 500 m from the OT [14]. These two additional constraints are illustrated in Figure 2.2 and have not been included in the analysis to find the best strategy to implement in the *driver*, however they will be included in Chapter 7.

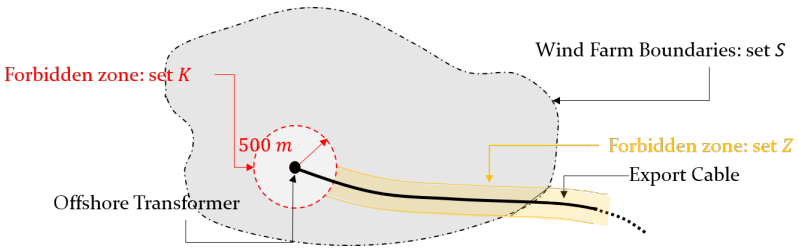


Figure 2.2: *Visualization of possible constraints to be added further. Minimum distance from the offshore transformer and forbidden zone around the export cable*

In the case described by Figure 2.2, the formulation of the additional constraints would look like[1]:

$$g_2(\mathbf{x}) = (x_i, y_i) \notin K \quad \forall i \in [0,1,...,N_T] \tag{2.9}$$

[1]They should be meant in the same form as Eqs. 2.6 and 2.7, with the exception that in this case they are *inequalities* (they should not belong to the "forbidden" polygon)

$$g_3(\mathbf{x}) = (x_i, y_i) \notin Z \quad \forall i \in [0, 1, ..., N_T] \tag{2.10}$$

being K and Z, respectively, the sets containing all the points in the design space which belong either to the transformer forbidden area or the restricted export cable zone.

2.4 Summary of the chapter

In this chapter, an outline on the offshore wind farm layout optimization problem has been provided. An optimization problem is made of three main ingredients: the objective function, which evaluates the performance of a solution to a design problem; the design vector, which gathers the variables which are allowed to change in the optimization; the constraints, which are limitations to the values the design vector can assume. If this theoretical overview is contextualized in the OWFLO, the objective function is the levelised cost of energy, the design vector is the layout of the wind farm and the constraints are some minimum spacing of $4D_{rotor}$ between the turbines and their location within the wind farm boundaries.

Chapter 3

The Python framework

3.1 Introduction to the chapter

This chapter presents an overview of the Python framework which has been used throughout this work. First of all, the concept of multidisciplinary design analysis and optimization (MDAO) is explained. A short description of WINDOW, the tool developed by Ph.D. candidate S. Sanchez Perez-Moreno is given, as well as its validation over known case studies, Eneco Luchterduinen and Prinses Amalia WindPark.

3.2 Multidisciplinary Design Analysis and Optimization (MDAO)

3.2.1 Overview about the MDAO workflow

To enhance the comprehension of this work, it is crucial to define the concept of MDAO (Multisciplinary Design Analysis and Optimization) workflow [1]. With respect to the common sequential design procedure, which optimizes the wind farm components *separately*, the MDAO analysis helps the user deal with the *interactions* between different design areas whilst *automating* the design process [1]. In other words, this approach couples the modules which analyze the different wind farm design areas - aerodynamics, support structures, cable topology etc. - with the aim of evaluating the cost and the overall performance *of the whole system*. The result is a design where all the parts are jointly optimized.
Within the MDAO domain, two components are identified. The former is called *analysis block* and it comprehends all the modular tools which refer to a specific physical/economic discipline; the latter is the *driver*, i.e. an optimizing procedure which calls the analysis block in each iteration [9].

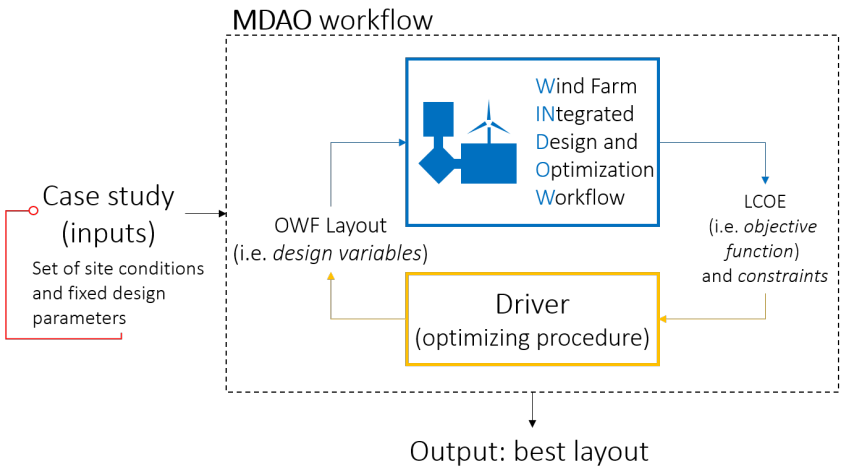


Figure 3.1: *The MDAO framework (simplified version)*

As can be seen in Figure 3.1, *WINDOW is the analysis block* which is repeatedly called, over the iterations, by the driver. The tool, which has been developed by TU Delft Ph.D. candidate S. Sanchez Perez-Moreno [9][1],

is a set of building blocks, each of them having a certain number of model fidelities, i.e. the available theories the user can choose from to describe a physical discipline[1]. On the other hand, the driver is the optimizing procedure, i.e. a combination of initialization (initial guess), constraint-handling techniques and algorithm. As mentioned in Chapter 1, the main purpose of this work is to develop an efficient strategy to be implemented in the driver.
Figure 3.1 provides a simplified overview of the MDAO workflow. Since in this work the use case is layout optimization with respect to the levelised cost of energy (LCOE), the LCOE of each layout is computed by WINDOW and used as an input for the optimizer; this generates a new layout which is again evaluated until a stopping criteria is met. The extended design structure matrix (XSDM) is shown below. The overall performance of the system is the objective function and the top-level layout optimizer calls the entire analysis block.

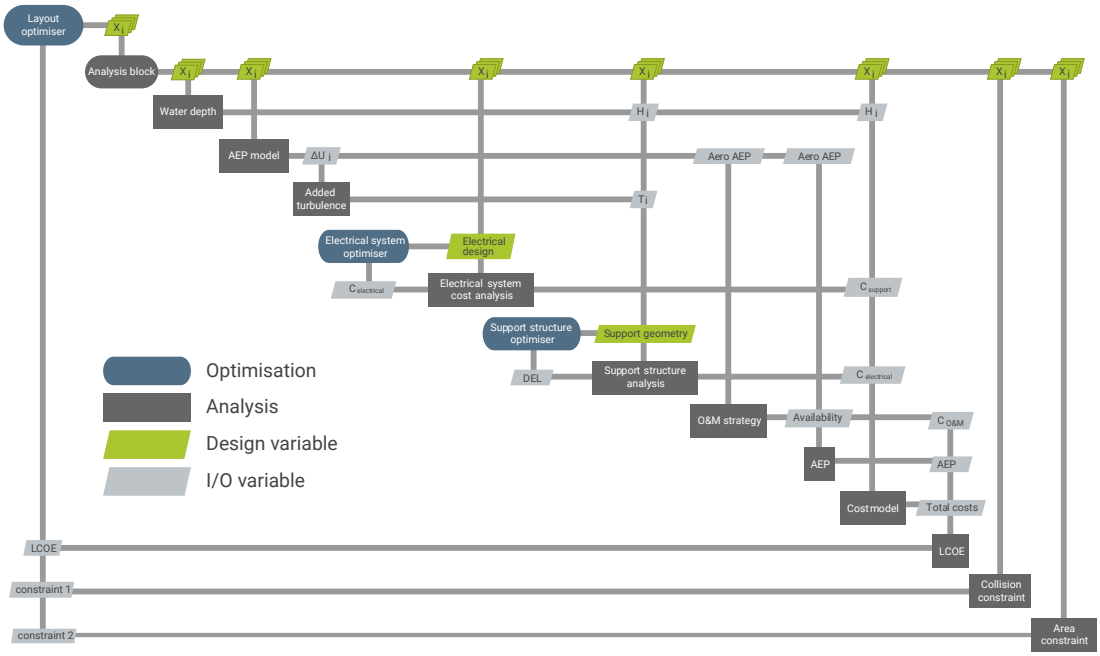


Figure 3.2: *The MDAO framework (detailed version - courtesy of S. Sanchez Perez-Moreno [1])*

Figure 3.2 shows the design areas which are included in the analysis block. As can be seen, the $[x, y]$ coordinates of the turbines - i.e. the layout - are the design variables (in green). These are the inputs for the Annual Energy Production (AEP) model, the Electrical system cost analysis, the Support Structure design, the overall cost model and the constraints. These, together with the LCOE, are returned to the layout optimizer. Since the Electrical model and the Support Structure design are stand alone nested optimizers - they return the best cable topology and the best monopile design w.r.t. the input layout - which require internal iterations, they are represented in blue[2]. The next paragraphs provide more detailed information.

3.2.2 The analysis block: WINDOW

WINDOW is made of several modules, each characterized by a certain number of available fidelity models. These are itemized in Table 3.1 [9].

Table 3.1: *Set of tools available in WINDOW for each discipline*

Module	Tools available
Downstream wake effects (for AEP)	Jensen[16], Larsen, Ainslie 1D, Ainslie 2D
Wake merging	Root sum square, Maximum deficit, Deficit product, Deficit sum
Wind turbine performance	Constant thrust coefficient and power (c_T and P curves), simple BEM
Wake turbulence	Constant turbulence, Frandsen [17], Danish recommendation [18] Larsen [19], Quarton[20]
Infield cable topology	Constant cost, Esau-Williams heuristic algorithm [21], Radial topology (POS) [22], Hybrid heuristic [23] algorithm
Support structure design	Constant support structure, TeamPlay[6]

[1]Example for the wake modeling discipline: Jensen, Larsen and Ainslie's theories are available
[2]Please note that they do not belong to the driver, but to the analysis block.

Almost all the models described in Table 3.1 are *low-fidelity* engineering models, meant to make decisions at an early design stage [1]. In compliance with [1], the chosen tools for each discipline are itemized below.

- **Downstream wake effects**. The Jensen model is the simplest wake model available and has been widely used in commercial software. The model neglects the wake region and assumes the turbulent wake to start after the rotor. However, the main assumption consists of the wake radius expanding linearly and a constant wind speed within the wake region [16]. According to Jensen, the expression which holds the velocity in the wake is written as:

$$v = V_0[1 - \frac{1 - \sqrt{1 - C_T}}{(1 + 2ks)^2}] \qquad [m.s^{-1}] \tag{3.1}$$

  where s (downstream distance) is equal to the ratio between the distance x and the rotor diameter $2R_0$, c_T is the thrust coefficient and a is the wake decay coefficient. According to Frandsen [24], k can be derived by the ambient turbulence intensity:

$$k = 0.4705 I_a + 0.004 \qquad [-] \tag{3.2}$$

  The value of I_a has been taken from the standards for the offshore environment [25]. The value is equal to 0.12. This gives $k \approx 0.06$

- **Wake merging**. This model was proposed by Jensen and Katic [16] and is based on the averaging of kinetic energy. In particular, it assumes that the kinetic energy deficit inside a mixed wake is the same as the sum of the energy deficits of each wake.

$$U_0 - U_i = \sqrt{\sum_k (U_0 - U_{k,i})^2} \tag{3.3}$$

  where U_0 is the undisturbed wind speed, U_i is the velocity at turbine i and $U_{k,i}$ is the wake wind speed of turbine k at turbine i.

- **Wind turbine performance**. At the moment of writing this thesis, WINDOW is being edited with a built-in RNA assembly optimizer. However, in this work the power curve and the thrust curve from the manufacturer have been used.

- **Wake turbulence**. The Danish Recommendation has been used. This calculates the turbulence added by the wake thanks to the mean wind velocity and the spacing amongst the turbines [18].

- **Infield cable topology**. The cable topology module is a built-in optimizer which deals with the so called OWFICTP (Offshore Wind Farm Infield Cable Topology Problem). According to Katsouris [23], the OWFICTP can be summarized by this statement: being the position of the turbines and the substation(s) known, "find the *optimal* inter-array cable topology which minimizes the total cable cost without violating the cable capacity". This means that given some input data, the cable topology module provides the best electric cable layout. The module requires the following input data: the placement of the wind turbines, the position of the substation(s) and the desired cable capacity. Regarding the last input, the user can choose up to three different integer numbers (a, b and c): these correspond to three distinct cable types which can connect at maximum a, b or c turbines. The most suitable current rating and cross section for each cable type are automatically computed and adjusted thanks to a database of real cables in WINDOW.
  The algorithm which is implemented was developed by Katsouris and it is a hybrid between two already existing strategies: a planar vehicle routing-based algorithm (named POS), developed by Bauer and Lysgaard [22], and a heuristic algorithm written by Esau and Williams [21]. From here onwards, Katsouris' methodology will be referred as *hybrid heuristic*. This approach is better performing than the other two when the offshore transformer is located *within* the wind farm boundaries, as it happens for all the case studies analyzed in this work. As Maselis shows [8], this algorithm is reliable and highly *sensitive* to the changing position of the turbines. These two aspects are definitely crucial when dealing with OWF layout optimization, as the position of the turbines is changed in each iteration.

- **Support structure design**. The support structure design module utilizes a built-in optimization to obtain the geometry of the monopile, transition piece, tower, and scour protection [1]. These quantities are calculated by taking the structural loads - occurring in selected *load cases* - as optimization *constraints*. In particular, the limit state for fatigue is neglected but is currently under research while this thesis is being written. On the other hand, the loads in the ultimate limit state are computed for a few load cases, by the means of a *static* analysis [6]. These load cases are defined in the standards [26] and are:

1. operation, U_{rated}, maximum wave height in 1-year extreme sea state;
2. parked, reduced gust in 50-year $U_{average}$, maximum wave height in 50-year extreme sea state;
3. parked, maximum gust in 50-year $U_{average}$, reduced wave in 50-year extreme sea state.

Safety factors are taken into account as well. These are also based on the regulations of IEC61400-3 [26], with the exception of a supplementary factor of 1.5 which is applied as a compensation for the neglection of the fatigue analysis.
All the input design variables are assigned values inside the loops which yield the outputs listed above. As already mentioned, the design variables inside the loop of the pile diameter, as well as the outputs, appear in the optimization constraints through the *structural loads calculation* (occurring at the load cases listed above): these are *aerodynamic* (calculated by discretizing the entire support structure), *hydrodynamic* (Morison's equation is used) and *gravity* (for the tower and TP wall thickness) loads.
The algorithm which is used in this module is not a proper optimization algorithm but it is Brent's *root-finding algorithm* [27], i.e. a procedure to find the root of functions [28]. The function to find the root of is the *stress factor* [6], which has been defined by Zaaijer as follows:

$$\sigma_{factor} = \frac{\sigma_{MAX}}{(\frac{\sigma_{cr}}{\gamma})} - 1.0 = 0 \qquad [-] \tag{3.4}$$

in which σ_{MAX} and σ_{cr} are, repectively, the maximum occurring stress in the structure and the yield stress of the material; γ is the overall product of the safety factors. The function σ_{factor} depends on the variables which have to be optimized: this function is interesting as it has a lower negative bound and a upper positive limit in the interval the design variables belong to. This is the reason why the root-finding algorithm suits well, as under these conditions convergence is guaranteed [28]. In other words, Equation 3.4 determines the optimal design variable by identifying the value at which the maximum stress equals the yield stress.

Other modules which are included in WINDOW are the Operation and Maintenance (O&M) model and the cost model. These are based on the extensive literature review performed by Zaaijer in 2013 [6].

- **The O&M costs**. No clear guidelines exist on how to correctly model the O&M costs [8]. Based on [6], an extremely simplified model is implemented. This computes the operation and maintenance costs as a constant value, equal to 6% of the magnitude of the AEP.
  It should be pointed out that Eneco is in possession of a tool, developed in MATLAB, which is based on Monte-Carlo simulations. However, this tool works satisfactorily when applied to already existing wind farms (whose measurements are known over the years) so some research about O&M costs *forecasting* still has to be carried out. A good example can be found in [3], where the author tries to investigate the effect of combining uncertainties (array efficiency and availability) on the energy yield of an OWF.
  Since the focus of this work is not on this topic, the simplified model described in [6] is adopted.

- **The cost models**. The cost models which are used consist of *investment costs* and *decommissioning* costs. The investment costs are divided into *procurement costs* (linear cost functions related to the mass of raw materials) and *installation costs*. The decommissioning costs are assumed to take place in the last year of the project lifetime and are split into *removal costs* and *disposal costs*. All the cost functions can be found in [6]

All the cost models are the result of an extensive literature study. They suit pretty well on a *preliminary phase of the project development* [11], but they do not take into account the complex interactions between shareholders. Moreover, some assumptions are made. Among those:

1. the use of a unique real interest rate r to all the products whereas, as stated in [6], it may have different values, each referring to a product.

2. the hypothesis of not discounting the investment costs as they are assumed to take place in year 0. In reality, there is a *timing* for the capital expenditures, i.e. a *misalignment* between the start of the expenditures and the energy selling. This spread of the investment costs before the start of the energy production is not taken into account.

3. the use of functions proportional to the mass for the evaluation of the procurement costs of raw material. In reality, these are subjected to variations due to fluctuations of the market.

Although these inaccuracies might contribute to the absolute results of the cost module, they are not likely to shift the value of the LCOE to a high extent.
However, if a more in depth implementation of the economic tool was applied, as well as a (raw) representation of the interactions among shareholders, then the background of the cost module would be likely to radically change. For example, the definition of the real interest rate which is used in the LCOE expression actually refers to the loans from the banks, which lend money to companies in order to develop offshore plants and which account for the highest percentage of the total investment in wind projects. In that case, if the goal is to model the discount of the combined costs from all the shareholders, the definition of weighted average cost of capital (WACC) [29] would seem more suitable. But in that situation, the current definition of LCOE should be modified accordingly. Carrying out such an elaborate analysis would be a very complex task whose influence on the layout optimization is not even proven. Therefore, to strengthen the economic background inside the WINDOW cost models, a good idea would be trying to at least adapt the procurement and the installation costs, as well as including the timing of the investment costs vs. the energy selling, filling the theoretical gaps with company-sensitive information which are normally not available on an academic level.

3.2.3 The driver: the optimizing strategy

As already mentioned before, the term *driver* refers to an optimizing routine which calls the analysis block (implemented in WINDOW) for a specific purpose. In this case, the purpose is *layout optimization*, the objective function is the LCOE and the constraints are the *spacing* between the turbines and the *boundaries* of the OWF.

The first research purpose in this thesis is indeed to efficiently design an optimizing procedure, i.e. the most desiderable combination of *initialization*, *constraint-handling technique* and *algorithm* to be evaluated by user-defined quantitative assessment criteria. This will be explained in detail in Chapters 5 and 6.

3.3 Presentation of the case studies

The two case studies which have been selected in this thesis both to validate of WINDOW and to assess the optimizing strategies are Eneco's offshore wind farms Prinses Amalia Wind Park and Luchterduinen.

3.3.1 Eneco Luchterduinen

Luchterduinen Offshore Wind Farm is located in the North Sea at approximately 23km off the coast between Zandvoort and Noordwijk (the Netherlands) and started to operate in May 2015 [30]. It consists of 43 Vestas V113-3.0 MW turbines, for a total installed power of 129 MW [30], and one offshore transformer. The layout of the WF is displayed in Figure 3.3 .

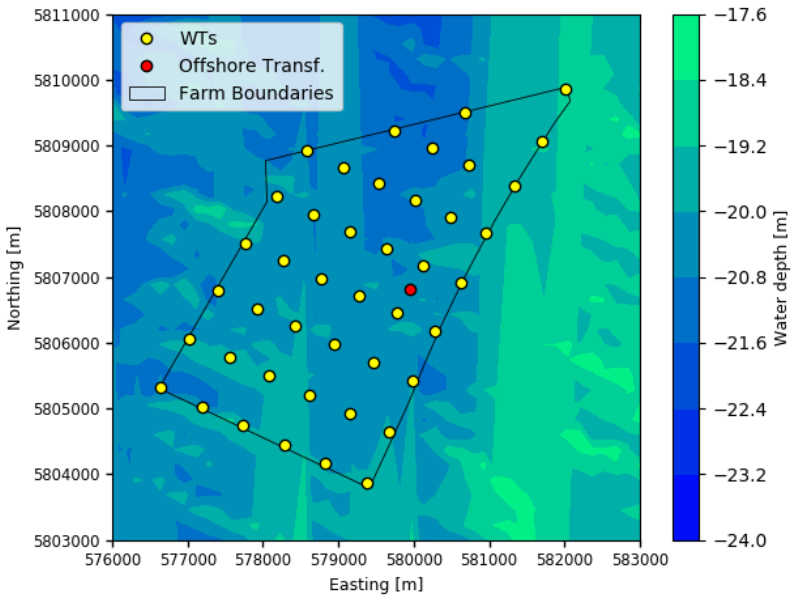


Figure 3.3: *Eneco Luchterduinen Wind Farm*

The farm specifications are shown in Table 3.2:

Table 3.2: *Luchterduinen Farm specifications*

Parameter	Symbol	Value	Unit of measure
Infield rated voltage	$V_{infield}$	33000	[V]
Transmission voltage	$V_{transmission}$	150000	[V]
Grid coupling-point voltage	V_{coupl}	380000	[V]
Total distance to grid	L_{total}	33	[km]
Distance to harbour (offshore)	$L_{harbour}$	25	[km]
Onshore transmission distance	$L_{onshore}$	8	[km]
N. of substations	$n_{substations}$	1	[-]

The wind turbine data are displayed in Table 3.3 [31]. The power and thrust curves are shown in Figure 3.4.

Parameter	Symbol	Value
Rated power	P_{rated}	3 MW
Cut-in wind speed	U_{cut-in}	3.0 $m.s^{-1}$
Cut-out wind speed	$U_{cut-out}$	25.0 $m.s^{-1}$
Rated wind speed	U_{rated}	12.0 $m.s^{-1}$
Rotor radius	R_{rotor}	56 m
Generator voltage	$V_{generator}$	657 V
Hub height (w.r.t. MSL)	z_{hub}	81.0 m
Front area nacelle	$A_{nacelle,front}$	$\approx 16\ m^2$
RNA mass	m_{RNA}	106000 kg
Maximum Thrust	T	524354 N

Table 3.3: *VestasV113-3.0 specifications*

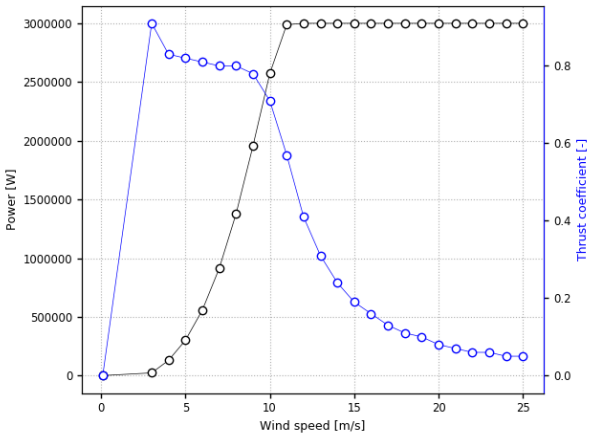


Figure 3.4: *VestasV113-3.0 P and c_T curve*

3.3.2 Eneco Prinses Amalia Wind Park

Prinses Amalia Wind Park (PAWP) is located in the North Sea at about 23km from the coast of IJmuiden [32]. It consists of 60 Vestas V80-2.0 MW turbines, for a total installed power of 120 MW. The farm started to operate in June 2008. The layout is shown below.

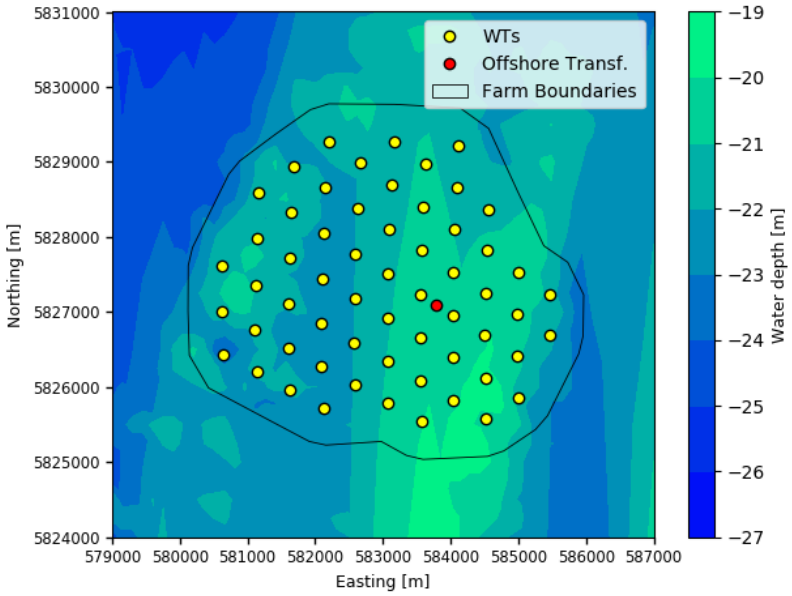


Figure 3.5: *Prinses Amalia Wind Park*

Table 3.4 displays the farm specifications.

Table 3.4: *Prinses Amalia Farm specifications*

Parameter	Symbol	Value	Unit of measure
Infield rated voltage	$V_{infield}$	22000	[V]
Transmission voltage	$V_{transmission}$	150000	[V]
Grid coupling-point voltage	V_{coupl}	50000	[V]
Total distance to grid	L_{total}	51	[km]
Distance to harbour (offshore)	$L_{harbour}$	28	[km]
Onshore transmission distance	$L_{onshore}$	23	[km]
N. of substations	$n_{substations}$	1	[-]

The wind turbine specifications for PAWP are illustrated in Table 3.5 [33]. The power and thrust curves are shown in Figure 3.6.

Parameter	Symbol	Value
Rated power	P_{rated}	2 MW
Cut-in wind speed	U_{cut-in}	4.0 $m.s^{-1}$
Cut-out wind speed	$U_{cut-out}$	25.0 $m.s^{-1}$
Rated wind speed	U_{rated}	12.0 $m.s^{-1}$
Rotor radius	R_{rotor}	40 m
Generator voltage	$V_{generator}$	680 V
Hub height (w.r.t. MSL)	z_{hub}	59.0 m
Front area nacelle	$A_{nacelle,front}$	$\approx 14\ m^2$
RNA mass	m_{RNA}	88500 kg
Maximum Thrust	T	475000 N

Table 3.5: *VestasV80-2.0 specifications*

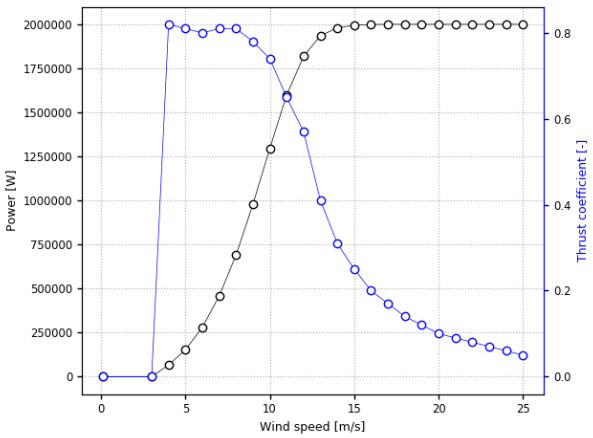


Figure 3.6: *VestasV80-2.0 P and c_T curve*

3.3.3 Input data for WINDOW - site conditions and fixed design parameters

The input data can be split into *environmental conditions* and *fixed design parameters*. The first category gathers all the information about wind conditions (Weibull distribution per wind rose direction sampling sector [34]), waves and current, bathymetry and soil conditions. The latter is a set containing all the fixed design parameters which are not modified by the driver. These are, for example, the turbine specifications (thrust curve and power curve, geometry and weight of the nacelle and rotor diameter), the coordinates of the offshore transformer (OT) and the distance between the OT and the grid coupling point onshore, the infield cable voltage and the transmission voltage. The fixed design parameters are presented in the previous paragraph thanks to Tables 3.2, 3.3, 3.4 and 3.5. The required environmental data are summarized in the table below.

Table 3.6: *Environmental input data*

Symbol	Type of environmental data	Unit of measure
α	Wind shear exponent	-
HAT	Highest astronomical tide	m
LAT	Lowest astronomical tide	m
Δz	Surge	m
$H_{s_{50-year}}$	50 year max significant wave height	m
$H_{s_{1-year}}$	1 year max significant wave height	m
$U_{current}$	Maximum current speed	m/s
d_{50}, d_{90}	Particle size distribution (soil)	m
ϕ	Friction angle (for soil assessment)	°
a, k	Weibull scale and shape factors	m, -
$N_{sectors}$	Number of wind direction sampling sectors	-
I_a	Ambient turbulence intensity	-
Bathymetry	Water depth	m

In this work, *the number of sectors which has been considered to sample the wind distribution is 8*. This means that each sector covers 45°. This choice has been made due to increasing computational cost when dealing with

a finer sampling. However, a study about the influence of the number of sectors on the final result has been carried out in Paragraph 6.4.2.

3.4 Validation of WINDOW

3.4.1 Introduction to the validation

As mentioned in Paragraph 3.2.1, the main goal of this work is to *identify the best optimizing strategy to be implemented in the driver* inside an MDAO workflow. As can be seen in Figure 3.1, the analysis block - WINDOW - and the optimizer exchange information about the *layout*, the *LCOE* and the *constraints*. However, they are *decoupled*, meaning that the user is able to modify the physical or the economic modules inside the analysis block if these are considered too rough[3]. Of course, the output from the optimization would be likely to shift if some parts of WINDOW were edited, but the performance of the optimizer *would be the same*, as this would not be affected by the changes in the analysis block.
Therefore, although this paragraph presents the validation of WINDOW over real case studies (PAWP and EL) to at least verify the ability of the tool to describe reality, this analysis is not meant to be extremely accurate. Unless too pronounced discrepancies occurred, the (foreseen) little misalignments between the model and the real data would be maintaned, as these would not affect the strategy in the driver, but only the final optimized result. In order to do the validation, the real layout and the environmental conditions were implemented in WINDOW and the results (according to the model) were compared to the real numbers. In particular, three areas were covered:

- the AEP;
- the support structure design;
- the infield total cable length.

Paragraphs 3.3.3 and 3.4.2 already dealt with the required input data for the validation.

3.4.2 Results and discussion from the validation

The validation of WINDOW has been performed by taking into account the three main design areas which characterize the design of OWF: the energy production, the support structure design and the cable topology. At this stage of the research, the environmental data and the fixed design parameters from PAWP and EL are used as input for the model. The results are compared to the real design of the two farms from Eneco's database. The table below shows the difference in percentage (for confidentiality reasons) between the output from WINDOW and the real case[4]. Since the support structure module provides a different geometry for each turbine (based on the environmental data), the lowest and the highest value are shown.

Table 3.7: *Results from validation of WINDOW over real case studies - PAWP and EL*

Discipline	Parameter	PAWP			EL		
		Real (normal. [%])	From model (normal. [%])		Real (normal. [%])	From model (normal. [%]))	
Aerodynamic module	AEP [Gwh/year]	100	-7.582		100	0.01	
Support structure module	$D_{pile_{bottom}}$ [m]	100	-2.5	5	100	-9.6	-7.6
	$D_{pile_{top}}$ [m]	100	-2.5	5	100	0.444	2.667
	$D_{TP_{bottom}}$ [m]	100	-8.531	10.189	\	\	\
	$D_{TP_{top}}$ [m]	100	14.201	37.574	\	\	\
	$D_{tower_{bottom}}$ [m]	100	14.201	37.574	100	0.444	2.667
	$D_{tower_{top}}$ [m]	100	0.259		100	-0.604	0.302
Infield cable topology	Length [km]	100	-26.667		100	-6.875	

From the table above, the following considerations may be drawn:

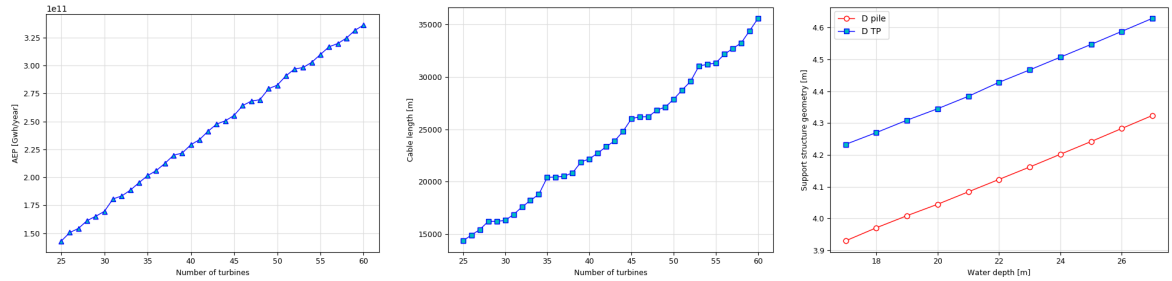
- the AEP module - based on the Jensen wake model and a root sum square wake merging - performs satisfactorily, especially for EL. Although a proper validation of the wake model would be desirable, in particular on the wake deficits per wind direction sampling sector over real measurements, this has not been done in this research. WINDOW allows the user to choose from higher fidelity wake models (such

[3] WINDOW only "sees" the layout from the driver and the driver uses only the LCOE and the constraints
[4] EL does not have a transition piece.

as Ainslie), which would have probably been more suitable to properly describe the aerodynamic losses in the wind farm. However, according to Daneshbodi [35], who studied the influence of the wake models on the layout optimization, although the results from different approaches differ from each other in terms of *magnitude of the energy yield*, the general patterns w.r.t. the wind direction sampling are the same. This means that a more accurate aerodynamic module would not shift the optimum. Since the Jensen model is analytic and computationally cheap, this choice was maintained.

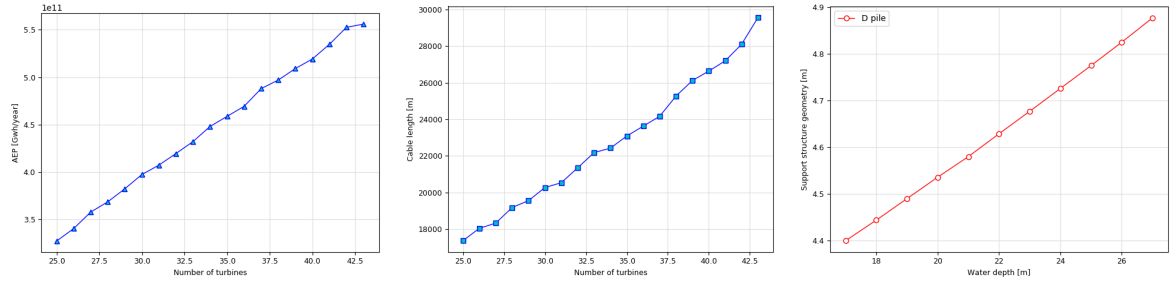
- The support structure module shows the biggest deviations from the real case. These can be explained by the way the support structure module has been developed. Both the pile and the transition piece (TP) have a cylindrical shape, meaning that the diameter remains constant. Moreover, the transition piece diameter linearly depends on D_{pile} due to knowledge based engineering rules (KBE) and is assumed to be equal to the tower bottom diameter. This means that if the TP is *conical* (as it happens for PAWP), significant misalignments take place, as shown in the table. However, the pile diameter (which is the parameter to be optimized) does not deviate too much from the real case (both in PAWP and in EL), therefore the results from the support structure design module have been considered satisfactory.
- The infield cable length shows clear discrepancies w.r.t. the real design. These can be justified by the fact that a built-in cable topology optimizer is present in WINDOW. This works on the basis of geometrical considerations (i.e. the distance between the turbines) and does not take into account some additional studies that may have led Eneco to increase the cable length during the design process.

In order to further validate the ability of WINDOW to model reality, some additional *fictitious* cases have been analyzed. Since the values in the table above are just single numbers, those results could have been determined by pure luck. Therefore, the sensitivity of the model needs to be verified by changing some input parameters whose consequent outcome was expected. This analysis helps the user realize that WINDOW is actually *sensitive* to different starting settings and gives results which are in compliance with reality. In particular, those fictitious cases were: AEP and cable length vs. variable number of turbines[5] and pile diameter vs. water depth. The results are shown below.



(a) *AEP vs. number of turbines - PAWP* **(b)** *Cable length vs. number of turbines - PAWP* **(c)** *Pile geometry vs. water depth - PAWP*

Figure 3.7: *Validation of WINDOW: sensitiviy of the analysis block to different input variables (PAWP)*



(a) *AEP versus number of turbines - EL* **(b)** *Cable length vs. number of turbines - EL* **(c)** *Pile geometry vs. water depth - EL*

Figure 3.8: *Validation of WINDOW: sensitiviy of the analysis block to different input variables (EL)*

From the graphs above, WINDOW's ability to *feel* changes in the input parameters is showcased. When the number of turbines increases, the AEP smoothly increases, as well as the cable length (as expected). The pile

[5]The turbines are added/removed *randomly*.

diameter is *almost* linearly increasing with the water depth[6]. Increasing the water depth leads to bigger bending moments in the pile due to a higher external loading and a larger lever arm.
Therefore, WINDOW is considered to be suitable to model the interactions between design areas for a preliminary estimation of future wind farm projects.

3.5 Summary of the chapter

To sum up, this chapter illustrated the concept of multidisciplinary design analysis and optimization (MDAO). An MDAO workflow is made of two components: the *analysis block* (called WINDOW), which provides the physical and economical description of a wind farm (and the consequent LCOE) based on its layout and the *driver*, i.e. the optimizer, which uses the computed LCOE to feed again the analysis block with a new layout. Many disciplines are covered within the analysis block and they all contribute to a satisfactory evaluation of the cost of energy. These are an aerodynamic module, a support structure nested optimizer (not to be confused with the *driver*), a built-in cable topology solver, an O&M module and a (rough) economic model.
The validation of WINDOW over real case studies provided good results, which do not deviate too much from the real design. This means that the tool is able to properly describe the interactions between different design areas in an OWF. The next step is therefore shifting the analysis towards the *driver*.

[6] If a fit is performed, it can be shown that the trend is not perfectly linear.

Chapter 4

The algorithms

4.1 Introduction to the chapter

In the previous chapters it has been mentioned that the *driver* is the component of the MDAO workflow which actually performs the optimization. Moreover, this is characterized by a combination of initialization, constraint-handling technique and, of course, algorithm. The algorithm is indeed the core of the driver, as it determines the sequence of operations to be followed to find the optimum. This chapter is entirely dedicated to the algorithms. Section 4.2 provides an overview of the suitable strategies for OWFLO; the next section presents an in-depth analysis of the selected algorithm; in Section 4.4, some tests over known functions have been performed.

4.2 Overview of the available algorithms

A large variety of optimization algorithms is described in literature and several classifications exist [36]. If the categorization is done based on the use of derivative of the objective function (or the gradient, depending on the size of f), then the algorithms can be listed into *gradient-based* and *gradient-free* [36]. The former, like Newton methods, use derivative information of f and are *deterministic*, i.e. under particular conditions the optimal point is guaranteed to be found without adding any element of randomness; the latter do not use any derivative information [36].
If on the one hand gradient-based algorithms are really efficient, they are computationally expensive; moreover, their reliability is valid only on a local level. In words, by using derivative information these methods have no ability to escape from local optima. This means that if the design space is very large, the objective function is nonlinear, maybe discontinous or multimodal and the problem is particularly complex, there is a high tendency to get trapped in a local optimal point, resulting in a poor exploration of the design space.
By contrast, gradient-free algorithms just use the value of f. In this work, deterministic gradient-free algorithms such as Nelder-Mead downhill simplex have not been considered; on the other hand, literature shows growing interest towards modern gradient-free, stochastic optimizing techniques, called *meta-heuristic* algorithms [36]. Methodologies with stochastic elements were often called *heuristic* in the past. Loosely speaking, *heuristic* simply means identifying a solution by trial and error [36]; in recent years, the prefix *meta* has been added to recognize those kinds of heuristic procedures which go *beyond* old heuristic methods, being smarter and better-performing. The main components of metaheuristic algorithms are *diversification* and *intensification* [36]. While the former means generating different solutions with the aim of exploring the search space on a *global* scale, intensification is the equivalent of focusing the search in a *local* region, being aware that a current good solution has been previously found in this region. A tradeoff between intensification and diversification should be accomplished to improve the rate of convergence. A wise choice increases the probability that solutions will converge to the optimum, whereas diversification by the means of *randomization* reduces the risk of falling into local optima while encouraging the diversity of solutions.

4.3 The selected algorithms

The algorithms which have been chosen for this research are meta-heuristic and are: Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Differential Evolutionary Algorithm (DEA). The reasons which con-

tribute to these choices are as follows. GA's and PSO's have already been widely used in offshore wind farm layout optimization problems [7][37][38][39][40]. By contrast, to the author's knowledge no examples on the utilization of the DEA applied to the OWFLO exist, with the exception of the paper by Rasuo and Begin [41], which however does not provide any clue on its actual implementation. Nonetheless, since many authors have stressed the ease of use and the capacity of the DEA to find good solutions [42][43], this approach has been tried. As will be clear further in the research, the evolutionary nature of the DEA has been considered very interesting by the author.
An important remark has to be stressed: since these algorithms are based on random search, there are good chances that the found solution will be close to the *actual* global optimum, but *there is no general evidence to demonstrate whether the final outcome is indeed the overall best* [36].

4.3.1 Genetic Algorithm

Among all the meta-heuristic models, genetic algorithms (GA) are the most used technique for wind farm layout optimization [44]. The pioneering application of this methodology in the wind energy sector was in the early 90's, thanks to the work of Mosetti et al. [7]. Generally speaking, GAs are *probabilistic* optimization strategies which take inspiration from Darwin's theory of evolution [45].
Genetic algorithms base their philosophy on the "survival of the fittest". In other words, an initial population of possible solutions is generated; each candidate solution competes with the others by being assigned a certain *fitness score*, computed by a fitness function f. The best individuals are selected to be the parents for the next generation. In complete analogy with Darwin's theory, these parents combine to create children (this is explained in more detail in the lines below): this mechanism is called *crossover*. The next generation is therefore made of fitter individuals than the previous one. The algorithm works until a certain convergence criteria is met: this might be a pre-defined number of iterations or a small difference between generation N and $N+1$ (tolerance) [45].
However, to prevent the algorithm from falling into a local optimum, *mutation* is modeled as well. Mutation is just a random deformation of the design variables in an individual, which occurs with a (usually) low probability.
Although Genetic Algorithms can deal with both *discrete* and *continous* design variables, due to their crossover-based nature these algorithms are particularly well performing mostly on *combinatorial optimization* problems, such as the famous *travelling salesman problem* [46]. The variables in such problems are discrete. In other words, the mechanism of offspring leads to good results as *it is able to find the best combination of the design variables*.
A more in-depth description of GAs is now provided. From the above text, five important phases are identified. These are reported below [47].

1. *Initial Population*: a population is nothing but a cluster of individuals. Every individual is a candidate solution. In Offshore Wind Farm Layout Optimization (OWFLO), this corresponds to a candidate *layout*. A set of parameters is used to characterize each individual. This set is a *gene* . In this analysis, each gene corresponds to a wind turbine. Genes are then joined into a string called *chromosome* (i.e. the individual itself). This means that every individual is described by its set of genes.
2. *Fitness Function*. The fitness function evaluates how fit an individual is. In this research, the fitness function *is* the objective function, i.e. the LCOE of the wind farm (as explained in Section 2.3.1). In minimization problems such as OWFLO, the lower the fitness value is, the fitter is the individual.
3. *Selection*. A user-defined percentage of individuals is chosen based on their fitness score.
4. *Crossover*. Crossover is the equivalent of natural reproduction: two parents combine their genes to create a child. In this thesis, each child inherits a gene - i.e. a turbine (x, y) position in the space - from either parent 1 or parent 2. Crossover can take place in different ways. Among others, these might be [45]:

   - N-point crossover. N points (with $N <$ chromosome$_{size}$) are randomly chosen. If $N = 2$ (two-point crossover), the first N genes in the child come from parent 1 and the last (chromosome$_{size} - N$) genes come from parent 2 or viceversa. This can be extended to $N > 2$. If $N > 2$, the child is divided into N "spots", each filled with the genes from either parent 1 or parent 2.
   - Segmented crossover. It is analogue to the previous one, but N varies at every generation.
   - Uniform crossover. Every child gene randomly comes from parent 1 or parent 2.

   The number of children to be generated is $N_{children} = size_{population} - N_{parents}$. This means that in every iteration the fittest individuals, i.e. the parents, directly go to the next generation. This approach, which is the same as the GA implemented in the software MATLAB [48], allows the algorithm *not to lose information about the previous optima*. In fact, if the population was totally replaced, the overall improvement of the population would not be guaranteed as it might happen for children to be less fit than their parents.

Several choices can be made to decide who each parent has to be combined with. In this research, to avoid a too pronounced similarity with one of the parents, all the possible combinations between parents are listed. The children are created by choosing random couples from the list of the permutations. The example in Fig. 4.1 is provided for some better understanding.

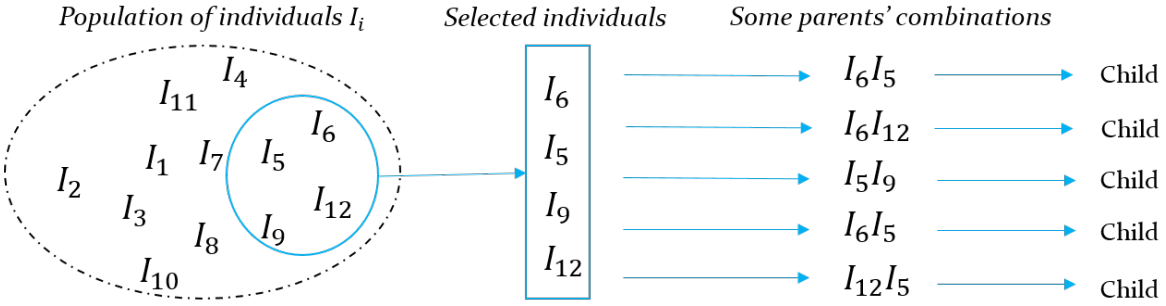


Figure 4.1: *Example on how the parents are randomly combined*

5. *Mutation.* Mutation is used to expand the search of the domain. Its probability of occurrence is set by the user and is usually quite low (below 15%) [45]: crossover must remain the main exchanging mechanisms among individuals. Other optimization algorithms, such as Differential Evolutionary Algorithm, are based on mutation rather than on crossover (Par 4.3.3).

The pseudo-code can therefore be summarized as follows:

Algorithm 1 GA

Initialise population; set crossover rate, mutation probability, percentage of fittest individuals, number of children
WHILE *condition is not met* **DO**:
  FOR each individual **DO**:
    assign fitness score by computing the objective function w.r.t. the individual
  END
  select the fittest individuals
  perform crossover
    IF *random() > CR* **DO**: % CR = 0.0001, quantity called "crossover rate"
      parents directly to new generation
      $n_{children} = population_{size}$ - $n_{parents}$
    ELSE:
      $n_{children} = population_{size}$
    END
  IF $random() \leq prob_{mutation}$ **DO**:
    perform mutation
  END
END

The user is able to set the following parameters:

1. the population size;
2. the percentage of selected individuals (usually 20-30 %);
3. the mutation probability;
4. the number of children.

4.3.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic, population-based methodology which was developed for the first time by J. Kennedy and R. Eberhart in 1995 [49] and is inspired by the movement of large birds' swarms. The strategy consists of a *population* of possible solutions, named *particles*, which at each iteration travel towards the best solution. To use a metaphor, a population can be thought of as a birds' swarm looking

for one piece of food in an area. Every bird is a flying particle which does not know where the food is, but knows how far it is in each iteration. At the beginning, this "distance" is a parameter to be set by the user. During a cycle, the distance of every bird from food is evaluated by a function called *fitness function*; all the resulting values are examined and the best *single* fitness value (i.e. lowest distance) over time is *remembered* by every particle. The *global* best result is the lowest one among the entire population. If in the subsequent iteration one particle scores a better fitness value than before, then its corresponding position in the search space is saved, otherwise it is discarded. Likewise, if a better global minimum is found w.r.t. the previous one, this value is taken as the global best. The swarm therefore is meant to *move* towards the minimum, as it is biased by the position of the best particle. However, to prevent the algorithm from converging too early, falling into a local minimum, an element of randomness is added to this research.

Going more in depth, two values are needed to describe the status of a particle in an iteration: the *position* vector, $\mathbf{p_i}$, and the *velocity* vector, $\mathbf{v_i}$ [49]. The former represents the position of particle i in the search space, while the latter determines the "rate of change" from the position at time t and the one at time $t+1$ [49]:

$$\mathbf{v_i}^{t+1} = C_0\mathbf{v_i}^{t} + C_1\epsilon_1|\mathbf{p_{best}} - \mathbf{x_i}^{t}| + C_2\epsilon_2|\mathbf{g_{best}} - \mathbf{x_i}^{t}| \qquad speed \tag{4.1}$$

$$\mathbf{x_i}^{t+1} = \mathbf{x_i}^{t} + \mathbf{v_i}^{t+1} \qquad position \tag{4.2}$$

In Equation 4.1, C_0 is a constant called the *inertia factor*, C_1 and C_2 are two constants named *learning factors* (usually equal); p_{best} and g_{best} are, respectively, the best local and global position ever found; ϵ_1 and ϵ_2 are a random number between 0 and 1 [49]. The pseudo-code might be thought as follows [50]:

Algorithm 2 PSO

initialise *velocity*, $\mathbf{p_{best}}$, $\mathbf{g_{best}}$, $\mathbf{x_{start_i}}$
WHILE *condition is not met* **DO**:
  FOR *each particle* **DO**:
    calculate fitness value
    IF *fitness value is better than* $\mathbf{p_{best}}$ **DO**:
      set current value as best local position
    END
    IF $\mathbf{g_{best_i}} \leq \mathbf{g_{best_{i-1}}}$ **DO**:
      Select the particle with the best fitness value as the new $\mathbf{g_{best}}$
    ELSE:
      keep old $\mathbf{g_{best}}$
    compute particle velocity and update it
    update particle position
  END
END

The user is able to edit the following parameters:

1. number of particles;
2. number of design variables;
3. the learning factors C_1 and C_2 and the inertia factor C_0;
4. the initial values, *velocity*, $\mathbf{p_{best}}$, $\mathbf{g_{best}}$, $\mathbf{x_{start_i}}$;
5. the stop condition, which may be $n_{iterations_{max}}$, the *tolerance* or a desired objective *value*.

The choice of the constant parameters, as well as a wise selection of the initial conditions, might exhibit a significant impact on the performance of the optimization [51]. For instance, a high value for the learning coefficient has the positive effect to let a particle explore a variety of even very different values, but the drawback is that it might be likely to change the design variables with a too fast rate, precluding any possible convergence. Some authors report that a correct parameter tuning can be performed by using a separate overlaying optimizer, or even fine-tuned during the iterations. However, in this work the selection of these values is the result of a *trial and error* process, as explained in Paragraph 5.2.2.

4.3.3 Differential Evolutionary Algorithm

The Differential Evolutionary Algorithm (DEA) is a strategy which shares several common features with Genetic Algorithms (GA). The idea behind these procedures is as follows: a *population* of potential solutions is set. During every iteration, the fitness of every individual is evaluated. The fittest individuals transmit their

information to the next generation by the means of *parents' crossover*. To prevent the algorithm from premature convergence, an element of randomness is added, which *mutates* some design variables in the individuals in order to explore the entire search space. The main difference between the already mentioned two procedures lies in the fact that GAs are mostly based on crossover, whereas DEAs rely on mutation. [42]. More precisely, the biggest contribution to the creation of a new generation comes from mutation.
Before getting more into the maths behind DEA, it is useful to provide the reader an overview of the pseudo-code [36]:

Algorithm 3 DEA

initialise *population size*, *design variables in each individual*, *fitness function (objective)*, *parameters*
WHILE *condition is not met* **DO**:
  FOR *each individual* **DO**:
    calculate fitness value and assign score (probability to be chosen) π
    perform mutation: compute the donor vector $\mathbf{v_j}$
    perform crossover: calculate the trial vector $\mathbf{u_j}$
    IF *fitness value(i) at generation G+1 is better than at generation G* **DO**:
      update fitness value(i)
    ELSE:
      fitness value(i) at generation G+1 is equal to fitness value at generation G
    END
  END
END

During the initialisation, the initial population size, p_{size} and the number of design variables are set. Every individual $\mathbf{x_j}$, with $j \in [1, p_{size}]$, is made of N design variables. The fitness function evaluates the goodness of every individual. In this case, since this work deals with *minimization*, *a lower value for the fitness function corresponds to a better individual*.
The algorithm works as follows. First, mutation of individual $\mathbf{x_j}$ occurs: [42]:

$$\mathbf{v_j^{G+1}} = \mathbf{x_{r1}^G} + F(\mathbf{x_{r2}^G} - \mathbf{x_{r3}^G}) \tag{4.3}$$

where $\mathbf{v_j^{G+1}}$ is the *donor vector* of individual j at generation $G+1$; F is the *scaling factor*, a number between 0 and 2; $\mathbf{x_{r1}^G}$, $\mathbf{x_{r2}^G}$ and $\mathbf{x_{r3}^G}$ are three *randomly* chosen individuals $\in [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{p_{size}}}]$ under the strict rule of being all different from each other and from x_j. The donor vector is therefore a combination of three randomly chosen individuals.
The following step is crossover:

$$\mathbf{u_j^{G+1}} = \begin{cases} \mathbf{v_j^{G+1}} & \text{if } rand(0,1) \leq \pi_j \quad \text{or} \quad j = I_{rand} \\ \mathbf{x_j^{G+1}} & \text{if } rand(0,1) > \pi_j \quad \text{and} \quad j \neq I_{rand} \end{cases}$$

in which $\mathbf{u_j^{G+1}}$ is the *trial vector*, I_{rand} is a random integral number $\in [1, p_{size}]$; π_j is the probability - i.e. the *score* - for the crossover to occur. The equation used to describe probability is set by the user. The author used the following expression:

$$\pi_j = 1 - \frac{fitness_j}{\Sigma(fitness_j)} \tag{4.4}$$

Every individual is assigned a certain fitness value. Since a lower magnitude of the fitness identifies a better individual (in minimization problems), the equation above ensures that the smaller $fitness_i$ is, the higher the probability of being chosen becomes. In the end, the selection takes place:

$$\mathbf{x_j^{G+1}} = \begin{cases} \mathbf{u_j^{G+1}} & \text{if } fit(\mathbf{u_j^{G+1}}) \leq fit(\mathbf{x_j}) \\ \mathbf{x_j^G} & \text{otherwise} \end{cases}$$

During this step, if the fitness value for the trial vector is better than the fitness value of individual j, then individual j updates to the trial vector value; by constrast, if no improvement takes place, the algorithm sticks to the previous $\mathbf{x_j}$ value.

4.4 Tests on known functions

Since GA, PSO and DEA play a crucial role in this research, some tests were carried out to verify the *good operation* of the coded algorithms. As already mentioned before, there is no overall evidence that the solution found

by means of meta-heuristics is the global optimum, however the ability of one algorithm to avoid falling into local minima can be evaluated by testing its performance over some functions, whose global optimum is known in advance. It is worth saying that this section is *not meant to assess the three chosen optimizing techniques*, as if one algorithm performed well on a test function, *there would be however no certainty about the goodness of the same procedure when applied to a particular engineering problem.* The so called "free lunch theorem", demonstrated by Wolpert and Macready in 1997 [52], states that if algorithm X works better than algorithm Y for some objective functions, then Y will outperform X for other functions. In other words, if averaged over *all the possible function space*, both procedures would perform well. The ability of the designer lies in figuring out what to apply when dealing with one design problem. Therefore, if the assessment of the algorithms was done at this point of the research, it would be useless when applied to OWFLO. The suitability of the GA, PSO and DEA has already been studied over the same known functions used in this paragraph [53]. Consequently, the tests illustrated below were carried out just to check the *correct operation of the coded procedures.* On the other hand, the assessment of the goodness of the same algorithms when dealing with OWFLO is presented in Chapter 6.

The functions which have been chosen to test the algorithms have been taken from literature [53]. These are Ackley's function, Rastrigin's function and Rosenbrock's function. All of these have been analysed in their $2D$ form (i.e. $f : \Re^2 \Rightarrow \Re$ [54]). The first two are characterized by a large number of local minima, whereas the last one is a popular test problem for gradient-based algorithms and therefore it is interesting to see what are the responses from the gradient-free strategies used in this thesis.

- **Ackley's function**
  Ackley's function is characterized by a hole corresponding to its global minimum and by a large number of local minima around it. The equation is:

$$f(x,y) = -20e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{0.5(\cos(2\pi x)+\cos(2\pi y))} + e + 20 \tag{4.5}$$

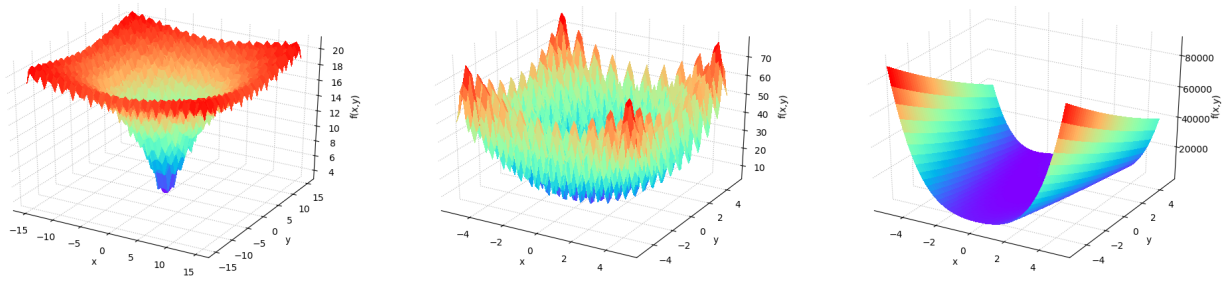
  The chosen domain has been put as $x \in [-15, 15]$ and $y \in [-15, 15]$; Figure 4.2a shows the function. The global minimum is located in $[0, 0]$. The value of Ackley's function in that point is $f(0, 0) = 0$.
- **Rastrigin's function**
  Rastrigin's function is also characterized by a large quantity of local minima. It is described by the following equation:

$$f(x,y) = 20 + (x^2 - 10\cos(2\pi x)) + (y^2 - 10\cos(2\pi y)) \tag{4.6}$$

  The chosen domain has been set as $x \in [-5, 5]$ and $y \in [-5, 5]$; Figure 4.2b shows the function. As Ackley's function, the global minimum is located in $[0, 0]$ and its value is $f(x, y) = 0$.
- **Rosenbrock's function**
  Rosenbrock's function is a popular test which has been widely used in the past to verify the goodness of gradient-based methods [53]. The minimum is $f(x, y) = 0$ and it is located in $[1, 1]$, in a narrow, parabolic valley. Although the valley is easy to find, convergence towards the *global* minimum is difficult. Rosenbrock's function is described by the following equation:

$$f(x,y) = 100(y - x^2)^2 + (x - 1)^2 \tag{4.7}$$

  As in Rastrigin's function, the chosen domain is $x \in [-5, 5]$ and $y \in [-5, 5]$; Figure 4.2c shows the plot.



(a) *Ackley's function* **(b)** *Rastrigin's function* **(c)** *Rosenbrock's function*

Figure 4.2: *Test functions*

The performance of the algorithms is reported in Tables 4.1, 4.2 and 4.3. The population size has been set as 30 in each case. The algorithms stop either when a tolerance of 10^{-3} between the values in subsequent generations is reached or when the maximum number of iterations (set as 800) is reached. The optimizations are, in all cases, *unconstrained*. The results are summarized in the following tables (each case has been run five times).

Table 4.1: *Result of the tests for Ackley's function*

	Genetic Algorithm			Particle Swarm			Differential Evolutionary		
Run	time 1 iter [s]	n. iter	result	time 1 iter [s]	n. iter	result	time 1 iter[s]	n. iter	result
1)	1.91	7	[0.00, 0.00]	1.49	179	[0.00, 0.00]	3.1	42	[0.00, 0.00]
2)	1.91	4	[0.00, 0.00]	1.49	165	[0.00, 0.00]	3.1	36	[0.00, 0.00]
3)	1.91	4	[0.00, 0.00]	1.49	173	[0.00, 0.00]	3.1	40	[0.00, 0.00]
4)	1.91	3	[0.00, 0.00]	1.49	145	[0.00, 0.00]	3.1	39	[0.00, 0.00]
5)	1.91	6	[0.00, 0.00]	1.49	180	[0.00, 0.00]	3.1	39	[0.00, 0.00]

Table 4.2: *Result of the tests for Rastrigin's function*

	Genetic Algorithm			Particle Swarm			Differential Evolutionary		
Run	time 1 iter [s]	n. iter	result	time 1 iter [s]	n. iter	result	time 1 iter[s]	n. iter	result
1)	1.88	3	[0.00, 0.00]	1.51	173	[0.00, 0.00]	3.16	73	[0.00, 0.00]
2)	1.88	2	[0.00, 0.00]	1.51	180	[0.00, 0.00]	3.16	65	[0.00, 0.00]
3)	1.88	3	[0.00, 0.00]	1.51	193	[0.00, 0.00]	3.16	67	[0.00, 0.00]
4)	1.88	3	[0.00, 0.00]	1.51	229	[0.00, 0.00]	3.16	70	[0.00, 0.00]
5)	1.88	2	[0.00, 0.00]	1.51	407	[0.00, 0.00]	3.16	64	[0.00, 0.00]

Table 4.3: *Result of the tests for Rosenbrock's function*

	Genetic Algorithm			Particle Swarm			Differential Evolutionary		
Run	time 1 iter [s]	n. iter	result	time 1 iter [s]	n. iter	result	time 1 iter[s]	n. iter	result
1)	1.85	3	[1.00, 1.00]	1.54	136	[0.99, 0.99]	3.35	50	[1.00, 1.00]
2)	1.85	4	[1.00, 1.00]	1.54	156	[1.00, 1.00]	3.35	44	[1.00, 1.00]
3)	1.85	3	[1.00, 1.00]	1.54	137	[0.99, 0.99]	3.35	44	[1.00, 1.00]
4)	1.85	2	[1.00, 1.00]	1.54	160	[0.99, 0.99]	3.35	48	[1.01, 1.00]
5)	1.85	3	[1.00, 1.00]	1.54	130	[0.99, 0.99]	3.35	52	[1.00, 1.00]

From Tables 4.1, 4.2 and 4.3 it can be seen that all the algorithms are able to find the global optimum. Although the "free lunch theorem" states that no evidence of the good performance of the algorithms might be given *a priori* even for the OWFLO problem, this analysis has shown *that the way these algorithms were coded was correct*. Therefore, potential failures during the optimization of a wind farm layout should be attributed to the "philosophy" behind the algorithms and not because of possible coding mistakes.
It is important to stress one last remark. The genetic algorithm failed one time to find the minimum for the Rosenbrock function. The problem was solved by raising the size of the population. As will be clear in Chapter 6, due to its combinatorial nature the GA is less likely to properly explore the design space and has therefore higher chances to fall into local minima.

4.5 Summary of the chapter

In this chapter, the algorithms which are used in the analysis of the *driver* inside the MDAO workflow are presented. These are meta-heuristic procedures and are: Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Differential Evolutionary Algorithm (DEA). To verify the goodness of the codes developed by the author, some tests over known challenging functions have been successfully performed.

Chapter 5

OWFLO problem setup

5.1 Introduction to the chapter

An in-depth overview about how different strategies for both the initial guesses, the constraint-handling techniques and the algorithms are employed in an OWFLO problem is given here. Section 5.2 illustrates the association between the algorithms and OWF's variables; Section 5.3 explains the different ways to initialise the OWF layout optimization; in the end, the constraint-handling techniques and the stopping criteria applied to OWFLO are presented.

5.2 Preparation of the algorithms for OWFLO

In this section, the parameters which have been set in the codes are summarized. Moreover, to help the reader understand how the OWF layout optimization problem is expressed in form of algorithm, the *chosen* correlation between the variables in the algorithms and the real-life variables in OWF design are illustrated.

5.2.1 Genetic Algorithm

In Section 4.3.1, a list of the required inputs and settings to run a genetic algorithm is reported. These are the population size, the design vector, the percentage of the selected individuals, the mutation probability and the number of children. The association between all of these and the OWFLO problem are illustrated in the table below.

Table 5.1: *Association between GA inputs and OWF parameters*

Genetic Algorithm inputs	**OWF design parameters**
Population size P_{size}	Number of different candidate layouts (initialized as explained in the previous section) which are used to compute the fitness function (every individual in the population is a layout)
Design vector $\mathbf{x}$	$[x, y]$ coordinates of each turbine. Dimension of $\mathbf{x} = 2N_{turbines}$
Percentage of selected individuals	The best layouts chosen to be parents ($30\ \% \rightarrow 0.3 \cdot P_{size}$)
Mutation probability	Probability of mutation of one layout (set as 13%)
Number of children ($= P_{size} - parents$)	The layouts created in the next generation ($0.7 \cdot P_{size}$)
Fitness function f	LCOE of the wind farm, computed by WINDOW. In this case, the lower the fitness is, the better is one individual

The percentage of selected individuals (i.e. the number of parents) has been set as 30%. Literature suggests using relatively low values, as in this way the evolution towards a fitter population is more likely to take place in a shorter period of time; however, if the number of parents is too small, there is the risk of getting trapped into local minima, as the algorithm would be too biased toward that small cluster of good individuals [45]. On the other hand, the mutation probability is usually very low (from ≈ 0.01 to 0.15). Even if a higher value would allow a broader exploration of the search space, too many mutants would make convergence difficult. Throughout the entire analysis, the *default* value of 0.13 has been used. There are several ways to implement

the mutation into a GA [45]. In this thesis, this has been defined as follows: up to 10% of the turbines *randomly* change their coordinates *as soon as they are within the boundaries and at least* $4D$ *from all the other turbines.* Thus, this kind of mutation introduces some element of "craziness" into the algorithm even though the mutating turbines are in feasible positions. This mechanism is the same among *all* the optimizing combinations involving the GA (including the grid-based approaches).
Regarding the number of children, i.e. the number of the layouts which are created from the parents' crossover, this is trivially equal to $(1 - 0.3) \cdot P_{size}$.
The population size is also a parameter to choose wisely. If it is too small, there is not much evolution to expect; on the contrary, if the population is too large, the computing time would be too extensive without ensuring a better quality of the results [36]. Some authors, like Pillai et al. [55] use a population size of 50; on the other hand, Wan et al. [39] use the value of 100. By contrast, in this thesis the value of 30 has been used both in order to ensure a fairer comparison with the other two algorithms, which use the same population size (as will be clear in Paragraphs 5.2.2 and 5.2.3), and to maintain the computational time within reasonable limits.

5.2.2 Particle Swarm Optimization

The following table summarizes the correlations between the variables in the PSO algorithm and the real variables in OWF.

Table 5.2: *Association between PSO inputs and OWF parameters*

Particle Swarm Optimization	OWF design parameters
Swarm size: P_{size}	Number of different candidate layouts (initialized as explained in the previous section) which are used to compute the fitness function (every particle in the population is a layout)
Design vector $\mathbf{x}$	$[x, y]$ coordinates of each turbine. Dimension of $\mathbf{x} = 2N_{turbines}$
Inertia/Learning factors C_0, C_1, C_2	Constant value (explained in main text)
Initial velocity	Constant value. (explained in main text)
Fitness function f	LCOE of the wind farm, computed by WINDOW. In this case, the lower the fitness is, the better is one individual

The swarm size has been put equal to 30. This value is used in the paper by Pillai et al. [55] and fits well with the computing power of the used PC. The paper written by M. Clerc in 2002 [56] and validated by Eberhart et al. [57] [40] highlighted a good choice for the inertia factor C_0 and the learning factors C_1 and C_2 to be resepctively 0.729, 1.494 and 1.494. These are therefore the default values which were used in the codes. The initial default velocity has been randomly set as $v \in rand[-1, 1]$; another way to initialize v is to put it equal to zero. No significant difference was noted in the analyses.

5.2.3 Differential Evolutionary Algorithm

As stated in Section 4.2, the DEA needs the following inputs: the population size, the design vector, the scaling factor F and the fitness function. The following table explains the correlation between these and the OWF design parameters.

Table 5.3: *Association between DEA inputs and OWF parameters*

Differential Evolutionary Algorithm inputs	OWF design parameters
Population size P_{size}	Number of different candidate layouts (initialized as explained in the previous section) which are used to compute the fitness function (every individual in the population is a layout)
Design vector $\mathbf{x}$	$[x, y]$ coordinates of each turbine. Dimension of $\mathbf{x} = 2N_{turbines}$
Scaling factor F	Constant value (explained in main text)
Fitness function f	LCOE of the wind farm, computed by WINDOW. In this case, the lower the fitness is, the better is one individual

The scaling factor F in literature usually assumes values between 0 and 2 [42], even if it is generally acknowledged that a scheme with $F \in [0, 1]$ is stabler [36]. The general value of F is problem-dependent and cannot be stated *a priori*. Several simulations were carried out to satisfactorily set this. From Equation 4.3 it can be easily noticed that a relatively high value of the scaling factor produces larger mutation amplifications [43]. The first trial which was done used $F = 0.5$, as suggested by [42]. This approach however made the algorithm diverge. This means that the DEA was not able to find a final solution as the rate of mutation was too high. The same

situation took place with $F = 0.4, 0.3, 0.2$. *The author identified the best values for F to be in the range between* $0.05 - 0.06$. In *all* the combinations (listed further in the report) where the DEA was used, these two values made the algorithm converge.

The population size was originally set as 16. On a later stage, that value has been set to 30. These numbers are in fact quite common in literature: J. Lampinen mostly uses those values and later moves up to a population size of even 50 to 120 individuals [43]. in this thesis, the value of 30 has been set as the maximum threshold in order to keep the time to complete one iteration within reasonable limits.

5.3 Initialization

5.3.1 Overview of the initializing techniques

Each of the algorithms presented in Chapter 4 has an *iterative* nature. Consequently, a *starting point* needs to be set. Several papers have been written about the importance of the starting point. Producing a good preliminary determination improves the optimizer performance and reduces the computational cost [58]. An interesting overview about the initialization in population-based optimization is shown in [59]. In this paper, the author illustrates the controversies around this problem and tries to systematically examine the effect of several initialization methods on the optimization of five benchmark functions; the conclusion is that some improvements may be expected when some more sophisticated initialization is employed. However, no scientific literature applied to OWFLO is available. On the other hand, a large number of articles on OWFLO has been published: each of them describes the way the design variables are initialized. The most widely used are listed below.

- Lückehe [60] et al., use random placement of the turbines, setting only the minimum and the maximum allowable values of the x and y coordinates. This methodology will be called "Random Initialization".

- Mosetti [7], Maselis [8], Emami [38], Wan [39] et al. use some *discretization of the design space*: the area of the wind farm is divided using a grid which prevents the turbines from violating any of the constraints. Therefore, the initial guess is always *feasible*. This approach will be named "Grid Initialization".

- Recently, a new approach is to use *heuristic* methods (see Paragraph 4.2) to place the turbines *randomly as soon as the resulting layout is feasible*. An example of this can be found in [61]. This strategy will be now renamed as "Smart Random Initialization".

Based on the list above, several initializing algorithms have been written and are presented in the following paragraphs.

5.3.2 Random initialization

The random initialization is the easiest way to set up an initial layout. It consists of the random placement of the turbines within a square area, which is characterized by the set $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$. Since the OWFs' boundary does not usually have a square shape, some turbines are likely to be outside the OWF area. Moreover, turbines can be placed very close to each other. The pseudo-code is as follows:

Algorithm 4 Random Initialization

FOR all the turbines **DO**:

$x_{turbine} = random([x_{min}, x_{max}])$

$y_{turbine} = random([y_{min}, y_{max}])$

END

A modification of this procedure is a random initialization which does not violate the boundary constraint but may still violate the spacing constraint. The idea is that this might "help" the algorithm and save some computational power. The pseudo-code then becomes:

Algorithm 5 Random Initialization - modified

FOR all the turbines **DO**:

WHILE $[x_{turbine}, y_{turbine}] \notin S$ **DO**:

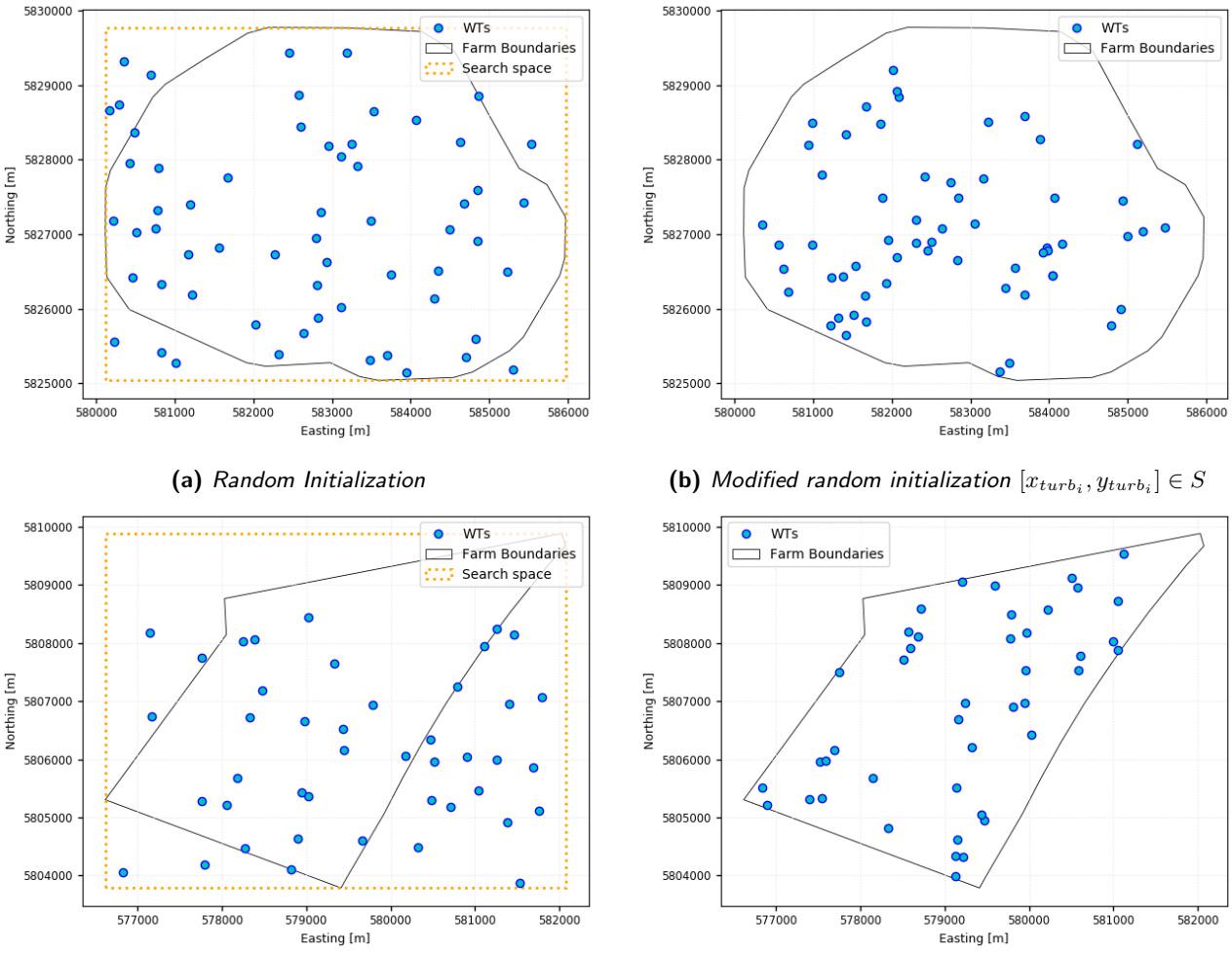
$x_{turbine} = random([x_{min}, x_{max}])$

$y_{turbine} = random([y_{min}, y_{max}])$

END

END

An example for both the situations is shown below for Prinses Amalia (Figs. 5.1a and 5.1b) and Luchterduinen (Figs. 5.1c and 5.1d):



(a) *Random Initialization*

(b) *Modified random initialization* $[x_{turb_i}, y_{turb_i}] \in S$

(c) *Random Initialization*

(d) *Modified random initialization* $[x_{turb_i}, y_{turb_i}] \in S$

Figure 5.1: *Random Initialization examples for PAWP (above) and EL (below)*

5.3.3 Grid initialization

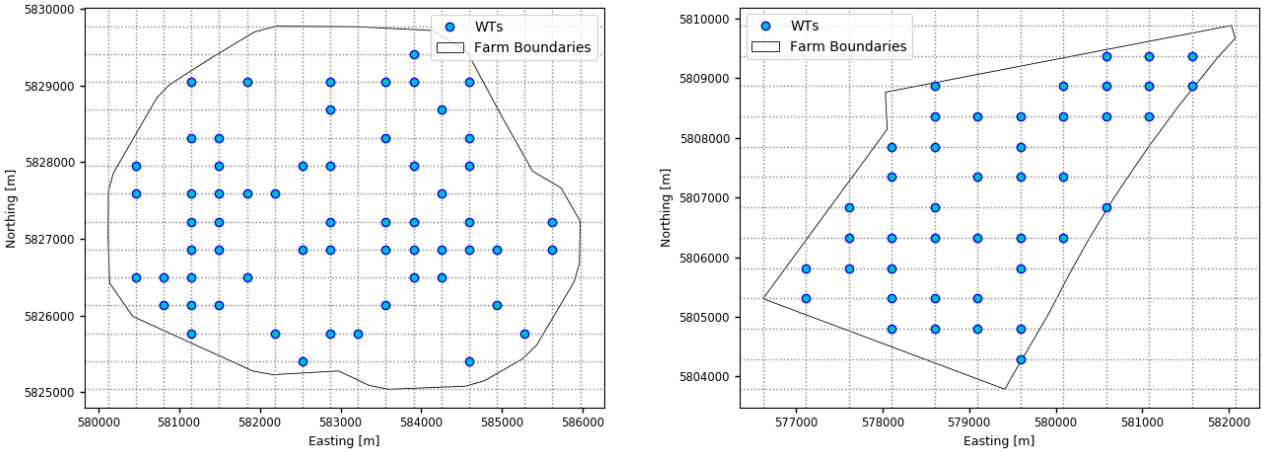
The grid initialization is a a methodology followed by Mosetti [7], Grady [37], Wan [39] et al. [38]. It is a simple and intuitive strategy: the design space is divided into square cells. One turbine can be placed only in the center of a cell (or on the vertices). In a grid-based approach, it is crucial to mention that if the turbines can only be put in fixed spots, then the problem becomes *combinatorial*, i.e. *find the best combination of turbines in the allowed places.* In Section 4.3.1, it is explicitly explained that genetic algorithms are suitable for these kinds of problems, because of their crossover-based way of exchanging information. In other words, *the coordinates of the turbines never deviate from the grid points (apart from mutation), because no summation nor subtraction of the* $[x, y]$ *coordinates takes place in a GA*. Therefore, in this thesis the grid initialization is coupled *only* with that optimizing strategy. This approach has two undeniable advantages towards the optimization:

- a significant amount of computational power is saved, due to the reduced number of allowable positions the turbines can assume [44];
- the constraints are automatically satisfied. In fact, the grid is meant to guarantee both the required minimum spacing between the turbines and their placement within the boundaries.

However, a big drawback is a relatively poor exploration of the design space. Wang et al. [62] investigated on the possibility of using an *equilateral-triangle mesh*. The resulting grid is finer and allows the optimizing algorithms to better explore the design space.
All the grids were developed by the author.

Square grid

The square grid initialization which has been adopted puts the turbines on the vertices of the cells.



(a) *Prinses Amalia* **(b)** *Luchterduinen*

Figure 5.2: *Square Grid Initialization. Example for PAWP and EL*

In practice, the procedure is: divide both the x and y axes by the *maximum* number of intervals that still complies with the spacing constraint. Then randomly place the desired number of turbines on the vertices of the resulting grid as soon as they are within the boundaries of the OWF. For the case studies taken into consideration, the number of allowed positions in the square grids is shown below.

Table 5.4: *Number of available spots in the square grid for PAWP and EL*

Wind Farm	Number of grid points in the square grid
Prinses Amalia Wind Park (PAWP)	165
Eneco Luchterduinen (EL)	72

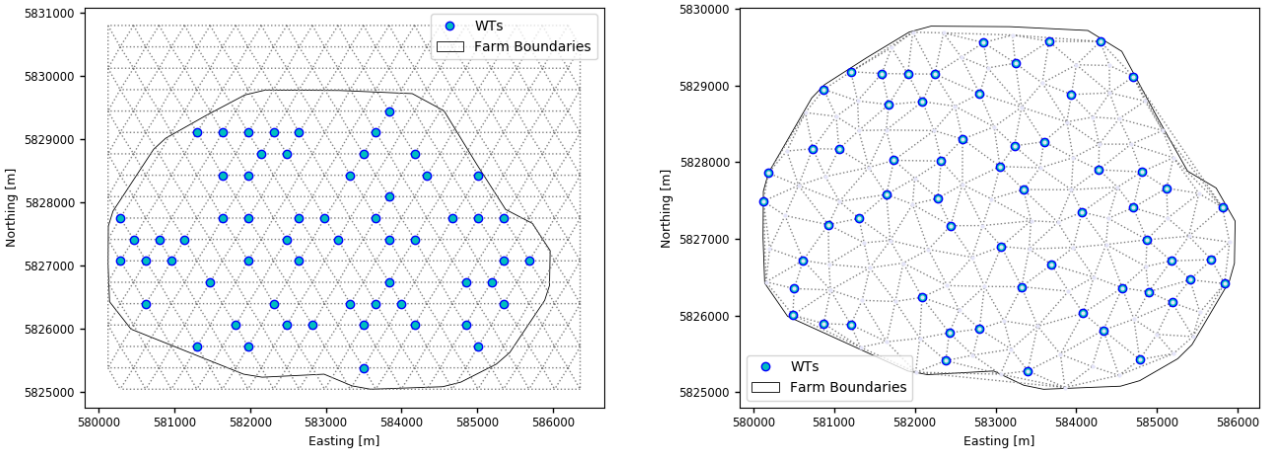
In this situation, the grid-creating algorithm is as follows:

Algorithm 6 Square grid creator

set: desired initial spacing (i.e. cell's side length), S (boundaries); initialize: $x = x_{min}, y = y_{min}, intervals_x = 0, intervals_y = 0$;
% Calculating the number of intervals to divide the x and y axes into
$intervals_x = int(\frac{x_{max}-x_{min}}{spacing}) + 1$ % int() provides an integer number
$intervals_y = int(\frac{y_{max}-y_{min}}{spacing}) + 1$
$x_{end} = x_{min} \cdot intervals_x$, $y_{end} = y_{min} \cdot intervals_y$
set allowable x, y *positions:* $x = linspace(x, x_{end}, intervals_x), y = linspace(y, y_{end}, intervals_y)$
WHILE $n_{turbines} <$ *desired number of turbines* **DO**: % Place randomly one turbine per time on one of the grid points
  WHILE $x, y \notin S$ *(i.e. the boundary of the WF)* **DO**: % Choose only the points inside the boundaries
    randomly place turbine in $x_i \in x$ *and* $y_i \in y$
    $n_{turbines} = n_{turbines} + 1$
  END
END

Triangled grid

The triangular grid increases the number of available spots w.r.t. to the square disposition [44]. In this study, *two* triangled grids have been analyzed: the *equilateral-triangle* grid [62] and the *random-triangle* grid.
To make the reader more familiar with these two concepts, first two illustrations are given below.



(a) *Equilateral-triangle grid* **(b)** *Random-triangle grid*

Figure 5.3: *Examples of Triangled grid initialization for PAWP*

The pseudo-codes of both these triangulations can be found in Appendix C. The reason why two triangular grids have been employed lies in the fact that both in the *equilateral-triangle* and in the *square* lattice the turbines are *aligned*. This means that the result from the optimization, being it good or bad in terms of the LCOE value, will follow the geometrical shape of the grid and some turbines will belong to the same "line". By contrast, an optimized solution initialised by means of the *random-triangle* grid, though being stuck to the geometry of the grid as well, will place the turbines without any alignment. The resulting layout would thus be *more* irregular. Therefore, *the two grids are designed in order to explore different parts of the design space*, i.e. some turbine dispositions which are captured in one approach are yielded in the other and vice versa. In particular, some remarkable differences are expected to take place from the annual energy production's point of view. Aligned turbines in fact feel the wake effect to a higher extent than misaligned ones.

The grid in Figure 5.3a is created in a similar way as the square lattice, but with some noticeable discrepancies. As further illustrated in Appendix C, the y axis is discretized in such a way that every *horizontal line* is drawn $\sqrt{3}$ times the spacing constraint (i.e. $L_{min} = 4D$) far from each other. This is because in every equilateral triangle the top vertex is $\frac{\sqrt{3}}{2} \cdot l$ – being l the side of the triangle – away from the basis of the triangle. Moreover, the x axis is discretized by $4D$, but every row is shifted $\frac{4D}{2}$ from the one above/below. This is clarified in the following figure:

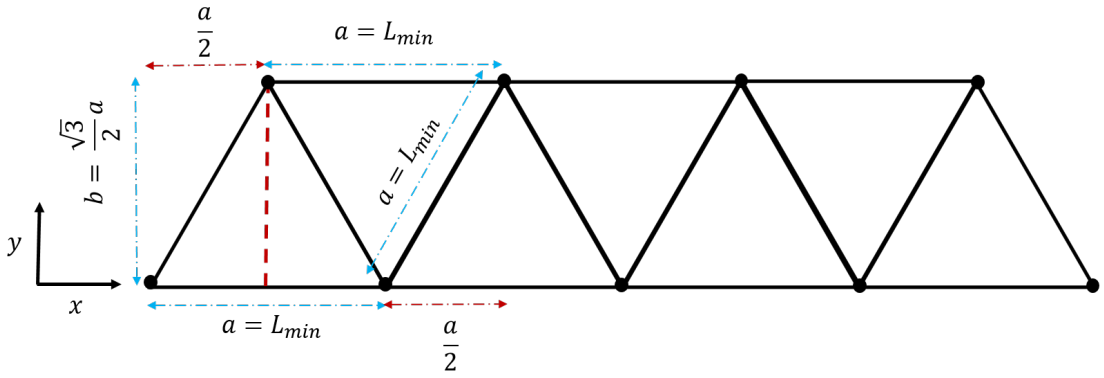


Figure 5.4: *Equilateral-triangle grid: a visualization. Every row is* $\frac{\sqrt{3}}{2} \cdot L_{min}$ *far from each other and the points in adjacent rows are shifted by* $\frac{L_{min}}{2} = \frac{a}{2} = 2D$

On the other hand, the grid in Figure 5.3b has been made as follows. The desired number of turbines has to be set, as well as the required spacing among them and a random number of spots n_{spots} (which has to be at least

equal to the number of turbines). The algorithm starts by drawing n_{spots} in such a way they respect *all* the constraints (i.e. at least 4D from each other and within the boundaries of the OWF). Once that all the spots have been drawn, more spots are added. In every iteration, one more spot is further inserted as soon as this is within the boundaries and far enough from all the previous spots placed that far. The algorithm stops creating new spots when no more points can be added (in this work, it has been supposed that no more points can be added if in one loop more than 3000 iterations take place). The triangulation which connects the spots is done with the Python package `matplotlib.tri`. Once the triangulation took place, the desired number of turbines is randomly placed on the grid points. This is explained more in depth in Appendix C. For the case studies taken into consideration, the number of allowed positions in the triangled grids is shown in Table 5.5.

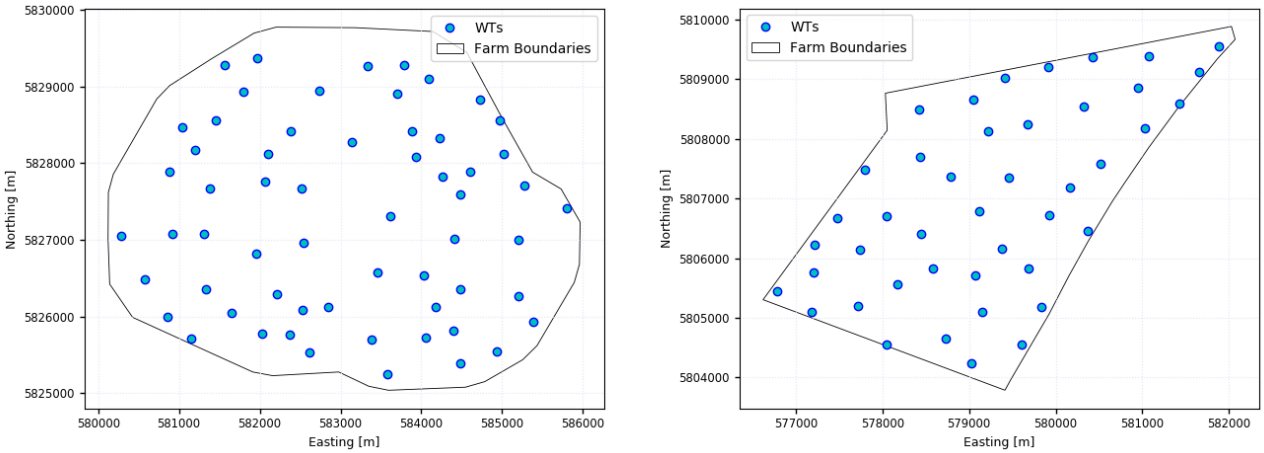
Table 5.5: *Number of available spots in the equilateral-triangle and in the random-triangle grid for PAWP and EL*

Wind Farm	Number of grid points in the equilateral-triangle grid	Number of grid points in the smart triangled grid
PAWP	185	146
EL	73	59

5.3.4 Smart random initialization

This type of random initialization uses randomness to create initial layouts. This approach allows the employment of *feasible* initial guesses *without constraining the turbines into fixed spots* (as in the Grid initialization).

The strategy the author implemented is as follows. The desired number of turbines has to be set, as well as the required spacing among them. The algorithm starts with a layout of 0 turbines. In every iteration, it adds one turbine as soon as this is within the boundaries and far enough from all the previous turbines placed that far. The algorithm stops when the desired number of turbines has been reached. This approach has been considered computationally reasonable by the author. In fact, another algorithm had been previously applied. This tried to place all the turbines *at once* until they complied with the constraints. That approach was too slow. On the other hand, adding one turbine per iteration is much faster. An example of this *feasible* initialization is displayed in Figs 5.5a and 5.5b.



(a) *PAWP* **(b)** *EL*

Figure 5.5: *Examples of Smart random initialization for Prinses Amalia (a) and Eneco Luchterduinen (b)*

The pseudo-code therefore is:

Algorithm 7 Smart random Initialization

set: spacing, boundaries (S), desired number of turbines
initialise $n_{turbines} = 0$
WHILE $n_{turbines} <$ *desired number of turbines* **DO**:
  WHILE *turbine violates distance constraint* **DO**:
    randomly place turbine in $x, y \in S$
  END
  $n_{turbines} = n_{turbines} + 1$
END

5.3.5 Conclusions on the initialization

To help the reader quickly visualize the main features of each initializing technique used in this work, the following table is provided.

Table 5.6: *Summarizing table of initializing techniques*

	Random	**Random modified**	**Square grid**	**Equilateral-triangle grid**	**Random triangle grid**	**Smart random**
Violates only g_1		•				
Violates g_1, g_2	•					
Feasible			•	•	•	•
Creates turbine alignment			•	•		
Eliminates the constraints from the optimization			•	•	•	

5.4 Constraint-handling techniques (CHT)

5.4.1 Overview of the constraint-handling techniques

In Paragraph 2.3.3, the constraints have been described. These are the minimum distance between the turbines (*spacing constraint* $g_1(\mathbf{x})$), equal to $4D_{rotor}$ [14], and the boundaries of the wind farm (*boundary constraint* $h_1(\mathbf{x})$). Several CHTs have been proposed throughout the years. A good summary was written by S. Koziel and Z. Michalewicz [63]. In this paper, the authors present three main methodologies to handle constraints in evolutionary algorithms, which are *penalty functions*, *methods which maintain the feasibility of the solutions* and *hybrid methods*. This last group of methods has not been taken into consideration in this thesis.
An excellent review can be found in [5]. The author here provides a good overview of constraint handling for heuristic algorithms within the OWFLO problem. Moreover, plenty of information can be found in [40], [64] et al [61].
Based on these notions, three main mechanisms have been chosen:

- *Penalty functions.* These strategies transform a constrained problem into an unconstrained one by adding a penalty to the objective function every time a constraint is violated.
- *Repair mechanisms.* These methodologies force every solution to be always feasible, i.e. it does not violate any of the constraints.
- *Absence of constraints.* As explained before in this chapter, these mechanisms ensure the optimization to be *unconstrained*, by the means of a *grid initialization.*

5.4.2 Penalty functions

The idea behind penalties in a *minimization problem* is to *add* a certain value to the objective function. In this way, the problem becomes *unconstrained* and the algorithm, at least theoretically, autonomously moves towards the feasible space [64]. Three different kinds of penalties can be adopted. The first one is *static*, i.e. it stays the same as long as the optimization runs; the second one is called *dynamic*, i.e. it is a function of the iteration number; the last one is named *adaptive*, i.e. it *adapts* to the problem, increasing or diminishing under certain conditions [64].
The penalty is usually designed in such a way it measures *how badly* the constraints are violated [5]. The most general form to write a penalty function is as follows [64]:

$$f(\mathbf{x}_i) = f(\mathbf{x}_i) + (\sum_{j=1}^{n} \sum_{i=1}^{N_{des.variab.}} C|g_j(\mathbf{x}_i)| + \sum_{k=1}^{p} \sum_{i=1}^{N_{des.variab.}} c|h_k(\mathbf{x}_i)|) \tag{5.1}$$

where $g_j(\mathbf{x})$ and $h_k(\mathbf{x})$ are, respectively, the magnitude of the violation of the inequality and equality constraints for individual $\mathbf{x}_i$; C and c are parameters chosen by the user; n is the amount of inequality constraints; p is the number of equality constraints. In this research, C has been assumed equal to c.

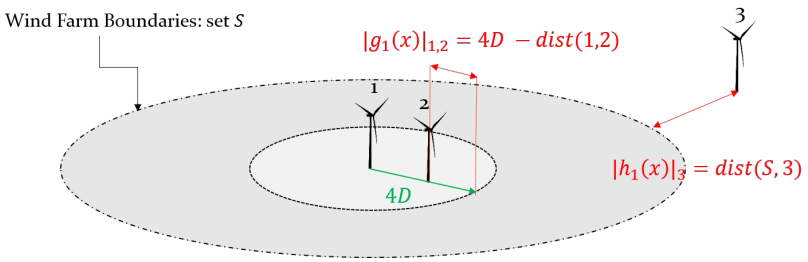


Figure 5.6: *Penalty function visualization: the magnitude of the constraint violation is added to the objective function, as shown in Equation 5.1*

A visualization is provided above: $|g_1(\mathbf{x})_{1,2}|$ is the distance that Turbine 2 violates the spacing constraint by; $|h_1(\mathbf{x})_3|$ is the point-line distance between Turbine 3 and the boundaries. A remark has to be stressed: the formulation of g_1 and h_1 in the penalty functions is slightly different than the one described in Paragraph 2.3.3. However, to avoid using a too heavy notation, these will be written with the same notation.
A clearer way of writing Equation 5.1 is provided below:

$$LCOE_i = LCOE_i + C(\sum_{turb=1}^{N_{turbs}} |g_1(\mathbf{x}_i)| + \sum_{turb=1}^{N_{turbs}} |h_1(\mathbf{x}_i)|) \qquad [c€.kWh^{-1}] \tag{5.2}$$

The C parameter is what makes the penalty approaches differ from each other, as explained in a few lines.
Before analyzing the three ways the penalties are applied to the OWFLO problem, one last crucial observation has to be done. As wisely argued by Runarsson and Yao [65], the use of these strategies in optimizing routines has the drawback of *increasing the roughness of the search space*. In words, when penalties are applied, the risk is that the algorithm focuses more on finding a *feasible* solution rather than the *optimal* one, falling into local minima. This will be discussed in more detail later in the report.

Static penalty functions

The static penalty function is the easiest way to assign a cost to an individual which does not respect the constraints. In this approach, the entire magnitude of the constraint violation is added to the LCOE. Since the violations are computed as distances, the unit of measure of C is c€$\cdot(kWh \cdot m)^{-1}$. Moreover, as the values of $|g_1(\mathbf{x}_i)|, |h_1(\mathbf{x}_i)|$ range from a few meters to even a few kilometers (order of magnitude: $10^0 - 10^3$) whereas the LCOE's order of magnitude is ≈ 10 (c€/kWh), in the static penalty approach *the feasible individuals always score a better fitness value than the infeasible ones*. That is why the factor C has been set to 1.
An approach which had been followed by the author was counting the *number of violations* rather than their magnitude. However, that methodology provided results of bad quality (which have not been reported) because the algorithms were not ready to distinguish, among two individuals with the same number of violations, which one was closer to respect the constraints. The equation which has been implemented in all the codes for the static penalty strategy is as follows:

$$LCOE_i = LCOE_i + (\sum_{turb=1}^{N_{turbs}} |g_1(\mathbf{x}_i)| + \sum_{turb=1}^{N_{turbs}} |h_1(\mathbf{x}_i)|) \qquad [c€.kWh^{-1}] \tag{5.3}$$

Dynamic penalty functions

A dynamic penalty is written similarly as the static function, but the C factor changes over the iterations. The general way to write a dynamic penalty is as follows [63]:

$$LCOE_i = LCOE_i + C(T)^{\alpha}(\sum_{turb=1}^{N_{turbs}} |g_1(\mathbf{x}_i)| + \sum_{turb=1}^{N_{turbs}} |h_1(\mathbf{x}_i)|) \qquad [c€.kWh^{-1}] \tag{5.4}$$

where T is the iteration number (C is a *function* of T). There are no general rules to correctly design the dynamic factor. However, a crucial guideline is that *in the first iterations, the algorithm should be able to explore the infeasible space without being penalized* [66]. In this way, at the beginning of the optimization, some infeasible individuals are capable of competing against feasible ones. This mechanism allows infeasible layouts early in the search, while continuously raising the penalty imposed, with the goal of eventually moving the final solution towards the feasible region [66]. The reason behind this methodology lies in the fact that some information

from the "bad" individuals might overall contribute to lowering the objective function more efficiently than in the static penalties.
After a trial and error procedure, a very well performing expression of C has been found by the author:

$$C = (0.0001 \cdot T) \qquad \alpha = 1.1 \qquad [c€ \cdot (m \cdot kWh)^{-1}] \tag{5.5}$$

This choice suits particularly well for the OWFLO problem because of the following reasons:

- The factor 0.0001 ensures that even a total violation magnitude of a few hundred of meters makes a layout competitive. As will be shown in the next chapter, in all the combinations which have been analyzed the biggest magnitude of constraint violation occurs during the first iterations. At that stage of the optimization, the value of the added penalty is still negligible. If the violations were too big, e.g. the turbines are too far from each other (increasing the cable cost) or too clustered (the wake effects decrease the energy production), the resulting LCOE *would not be competitive by itself*; however, the value of 0.0001 allows layouts with a relatively *low* violation magnitude to be chosen for the next iteration. Indeed, if one individual is characterized by a "good" turbine disposition even if the constraints are not respected, the information which in the case of the static penalty would be lost are, instead, saved.
- The C factor is designed in such a way it grows relatively fast, preventing the population from getting entirely infeasible. As will be illustrated in the results, the dynamic penalty function will be only coupled with a feasible initialization. It is essential to always maintain some feasible individuals inside the population. Because of the reasons explained in the previous point, some infeasible layouts may overcome the feasible ones. The expression in Equation 5.5 ensures a relatively fast growth of the penalty, preventing the population from becoming all infeasible.

Adaptive penalty functions

The adaptive penalty function approach was first introduced by Bean and Hadj-Alouane in 1992 [66]. It consists of an increase or a decrease of the penalty function as long as the optimization runs, based on criteria set by the user. Since the only authors who tried to apply the adaptive penalty function to the Wind Farm Layout Optimization problem (onshore) are Lückehe et al. [60], their approach has been used. This is as follows: if, after a certain number of iterations, less than 20% of the individuals in a population is feasible, then the penalty is increased, otherwise it is decreased. In this work, the expression for the adaptive penalty has been written as follows:

$$LCOE_i = LCOE_i + \lambda \cdot C(T)^{\alpha}\left(\sum_{turb=1}^{N_{turbs}} |g_1(\mathbf{x}_i)| + \sum_{turb=1}^{N_{turbs}} |h_1(\mathbf{x}_i)|\right) \qquad [c€.kWh^{-1}] \tag{5.6}$$

in which λ is the adaptive constant (note: $\lambda_{start} = 1$):

$$\lambda = \begin{cases} \lambda + 50 & \text{if less than 20\% of the population is feasible} \\ \lambda - 50 & \text{otherwise; if } \lambda = 1\text{, do not subtract further} \end{cases}$$

Equation 5.6 is the same as Equation 5.4, but the C constant is different. To sum up, the expression in Equation 5.6 consists of a *dynamic part*, i.e. C, and an *adaptive part*, i.e. λ. The parameter C is one order of magnitude lower than the one in Equation 5.5, i.e. 0.00001. The reason is because in this way the competition between infeasible and feasible individuals *would last for more iterations, ensuring even a better exploration of the infeasible space*. However, the λ parameter acts as a back-up mechanisms which restores the population over the course of the iterations, ensuring a minimum number of feasible layouts.
A crucial remark has to be mentioned. In order to have an efficient adaptation, the challenge is to have a dynamic part which does not grow too fast and an adaptive which intervenes when necessary. After having performed a large number of tests, three cases were identified to take place:

- if the dynamic part increased too fast, there would be no difference between an adaptive and a dynamic penalty. In fact, the population would never be made of a too large number of infeasible individuals, making λ useless.
- If C increased too slowly, then the output would *fluctuate*, as the penalty mechanism would entirely depend on λ. Therefore, as soon as a minimum number of feasible individuals is present in the population, the optimization runs almost unconstrained because C is very low. If too many infeasible individuals started to appear in the population, then the penalty would be increased by the λ factor until at least 20% of the candidate solutions is feasible; but, when this happens and the population is restored, λ *decreases* by the

same quantity and the optimization would be, again, almost unconstrained (C is still too low), resulting in the same starting situation. Consequently, it would be extremely difficult to make the population converge towards a non-violating solution [1].

- If C increases with a good rate (small enough to allow infeasibility to be present in the population), when a situation in which λ increases and then decreases again takes place, *C has already increased by a small percentage in the meantime, which is however enough not to have an "almost unconstrained" situation.* Fluctuation would be avoided [2].

Obviously, another way to design adaptives is deleting the dynamic part and acting uniquely on λ, making sure that the increasing rate is not the same as the decreasing rate. However, this approach was judged even more difficult by the author. In fact, it is unclear to what extent the difference between increasing and decreasing λ should be pronounced; moreover, it is hard to design a λ value which avoids fluctuation and that, in the meantime, allows a progressive rejection of infeasible solutions over the course of optimization.

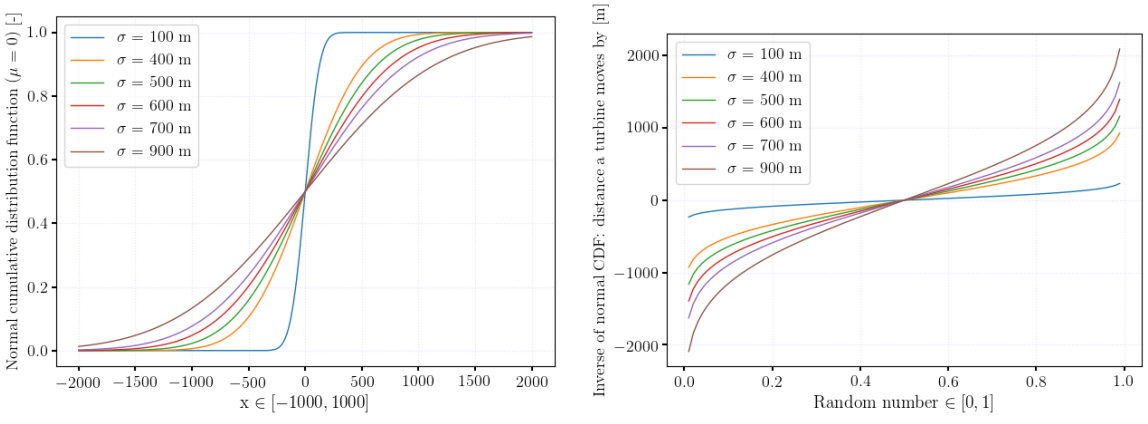
5.4.3 Repair mechanisms

The repair mechanisms are strategies which force every candidate solution to be, over the course of the optimization, feasible. This methodology has been adopted in particle swarm optimization [67] [9], in random search algorithm (not treated here) [68] and in evolutionary algorithms [69].
These may be gathered in two groups: *resampling* and *repairment*. The former creates a new, feasible layout every time a violation occurs; the latter changes the position of the turbines violating the constraints, keeping the old placement of the "good" ones [5]. The first approach has the disadvantage of ignoring the progressive knowledge acquired over the optimization and has therefore been discarded. The second strategy was therefore applied. Two techniques were tried:

- The bad turbines are replaced randomly. Several attempts were done to test this strategy but in none of the trials the algorithms were able to converge. The reason was probably that, especially at the early stage of the optimization, the number of turbines to be replaced is high and a random positioning cancels all the information about what the layout looked like, inhibiting the convergence.
- The bad turbines are adapted by sampling from a normal distribution [70] [71]. In this approach, a turbine keeps randomly changing its $[x, y]$ coordinates sampling from the inverse of a normal cumulative distribution function until it finds a feasible spot. The mean μ (in this case equal to the original x or y coordinates of the turbine) and the standard deviation $S.D. = \sigma$ need to be set *a priori*. Figs 5.7a and 5.7b show how the normal sampling is performed. The normal cumulative distribution function (on the left) is computed as follows:

$$CDF_{normal} = \frac{1}{2}[1 + erf(\frac{X - \mu}{\sigma\sqrt{2}})] \tag{5.7}$$

where erf is the error function and X is the independent variable; the inverse of the CDF is the normal sampling distribution function, which indicates, for each standard deviation, to what extent the windmills can change from their original positions when a random number $\in [0, 1]$ is chosen.



(a) *cdf* **(b)** *cdf* $^{-1}$

Figure 5.7: *Steps in a normal distribution*

[1] What the user wants to achieve in the end is a *feasible* solution
[2] The graphs showing the trend of the overall constraints' violations over time are shown in the next chapter

The expression used to draw the image on the right is as follows [3] [71]:

$$CDF_{normal-inverse} = \mu + \sigma\sqrt{2} \cdot erf^{-1}(2 \cdot rand[0,1] - 1) \qquad [m] \tag{5.8}$$

A turbine can move only within a certain area and has larger probabilities to be shifted by a relatively small number (Figure 5.7b). As mentioned before, the value of the mean is equal to the x (or y) coordinate of the turbine. The standard deviation has been set as 400 m. The reason is because with $\sigma = 400m$ the turbines have enough space to move *without being placed too far away from their original position*, as can be seen from the graphs above. This approach made all the combinations converge to a solution.

5.4.4 Absence of constraints

A common approach to handle constraints within the OWFLO problem is to discretize wind farm area using a grid which prevents all the turbines from violating any of the constraints [5]. Several authors used this approach, which has been mostly combined with genetic algorithms [7] [37] [38] [39] [72]. The more the constraints are, the more it is difficult to build a grid. However, this has to be done just for the initialization, as explained before in this chapter. It can be concluded that the grid is both an initializing methodology and, at the same time, a CH technique.

5.4.5 Conclusion on CHTs

The following table is provided to give an overview of the constraint-handling techniques used in this work.

Table 5.7: *CHTs summarizing table*

Constraint-handling technique		**Values of the parameters**	
Penalty function	Static	$\lambda = 1$	$C = 1$
	Dynamic	$\lambda = 1$	$C = (0.0001 \cdot T)^{\alpha}$ $\alpha = 1.1$
$(LCOE = LCOE + \lambda \cdot C \cdot g_{(}\mathbf{x}))$	Adaptive	$\lambda = \pm 50$	$C = (0.00001 \cdot T)^{\alpha}$ $\alpha = 1.1$
Repair mechanism		Normal distribution sampling: $\sigma = 400$, $\mu = x$ (or y)	
Absence of constraints		Grid initialization	

5.5 Stopping criterion

The stopping criterion which has been considered is as follows. The algorithm stops when all the individuals of the population differ from less than a certain tolerance between subsequent generations. This tolerance is a 10 m difference between the position of the turbines of the individuals.

5.6 Summary of the chapter

In this chapter, the way the OWFLO problem is coupled to the algorithms, the initializions and the constraint-handling techniques is explained. The population of individuals which is needed to start the algorithms is nothing but a large array which contains the *candidate layouts* (i.e. the individuals), each of them made of the same number of turbines. This population can be initialized either via a random methodology, which places the turbines in the OWF area without paying attention to the constraints, or via a grid-based approach, which forces the turbines to belong to pre-defined spots, or via a feasible initialization, which yields only feasible starting layouts.
The constraint-handling techniques which have been explored were: penalty functions (static, dynamic and adaptive), which simply add a penalty to the objective function (to be minimized) in order to bias the algorithm towards feasible solutions; repair mechanisms, which force the population of candidate layouts to be always feasible over the iterations; absence of constraints, which is the mechanism adpoted by the grid-based initializations.

[3] Equation 5.8 is also called *quantile function*

Chapter 6

Results

6.1 Introduction to the chapter

In this chapter, the results from the optimizations are presented. First of all, an overview of all the combinations of the optimizing strategies is provided, as well as the assessment criteria which are used to rank them. Secondly, the best three combinations are identified. The optimized results for PAWP and EL are then compared to the performance of the real design. In the end, two additional studies are carried out: the former investigates on the effect of the wind direction sampling on the optimization; the latter gives insights on the influence of the design areas on the optimization and the potential contributions of those disciplines to the LCOE improvement.

6.2 Analysis of the combinations

In this section, an overview of all the analyzed combinations is given; after that, the motivations behind their choice and the trends which characterize them are explained.

6.2.1 Overview of the combinations

Table 6.1 summarizes all the combinations which were tested.

Table 6.1: *Summary of all the analyzed combinations of algorithm-CHT-Initialization blocks*

	Algorithm			**Constr. Handl. Technique**				**Initialization**					
	GA	PSO	DEA	Static Pen.	Dyn. Pen.	Adapt. Pen.	Repair	Rand.	Rand. modif.	Equil. Triang.	Rand. Triang.	Square	Feas.
1	•			•									•
2	•			•				•					
3	•			•					•				
4	•				•								•
5	•					•							•
6	•						•						•
7	•									•			
8	•										•		
9	•											•	
10		•		•				•					
11		•		•					•				
12		•		•									•
13		•			•								•
14		•				•							•
15		•					•						•
16			•	•				•					•
17			•	•					•				•
18			•	•				•					
19			•	•									•
20			•		•								•
21			•			•							•
22			•				•						•

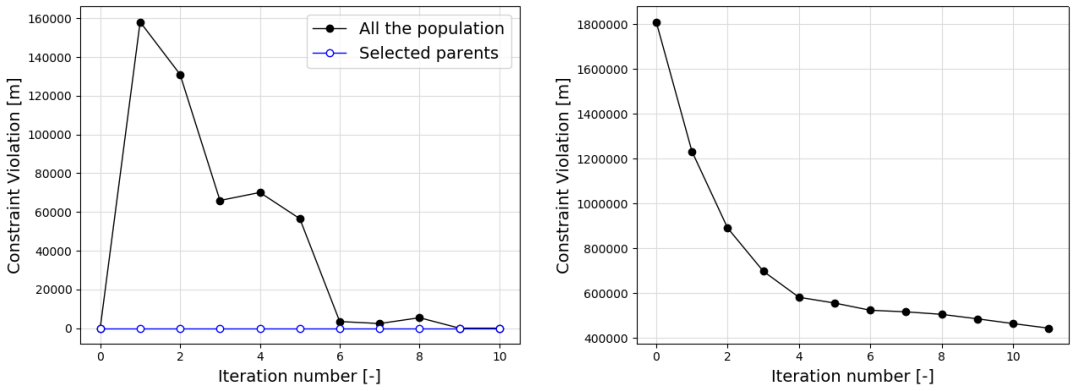
It is important to note that the 16_{th} and 17_{th} block consist of a *hybrid initialization*, i.e. two initializing mechanisms are chosen to create one population. The reason is explained in the next paragraph.
All the combinations were run from *five to seven times*.

6.2.2 A closer look at the combinations

As can be seen from Table 6.1, three main groups can be identified. These are the GA, PSO and DEA algorithms, each coupled with a different initialization and constraint-handling technique. The following lines try to explain *why* those combinations were selected and which are the patterns which characterize them.

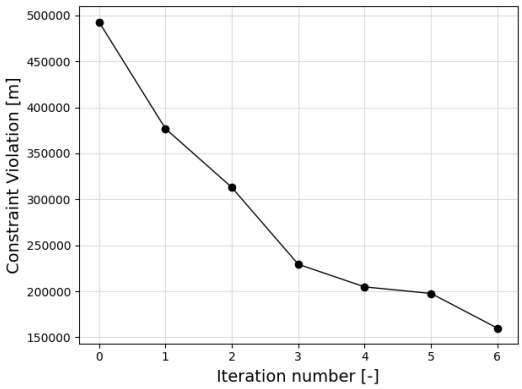
Genetic Algorithms

The first three blocks (1, 2 and 3) have been chosen to investigate on how the *static penalty functions*, which are the most widely used in literature, work when coupled with different initializations. The graphs illustrating the total constraint violation magnitude (in m) of the population vs. the number of iterations are shown below.



(a) *Feasible Initialization - Combination 1*

(b) *Random Initialization - Combination 2*



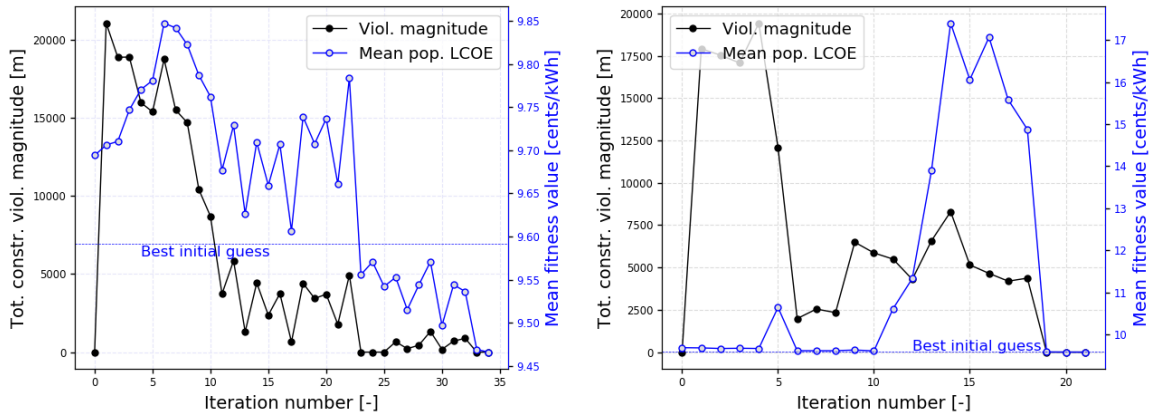
(c) *Random modified Initialization - Combination 3*

Figure 6.1: *Examples of the constraint violation plot for the genetic algorithm with static penalty (EL). The blue trend in Figure (a) indicates that in every iteration the selected parents are feasible*

As can be seen, the only satisfactory approach is combination a, as the others fail to find a feasible solution. When a GA, initialized with a feasible population and a static penalty, selects the best individuals and then combines them by means of crossover, then all the children which violate the constraints cannot compete with the parents (which are all feasible). However, since the crossover mechanism is performed randomly, *there might be some children which do not violate any of the constraints*. This is because it may happen for one of the two parents to put its genes (i.e. the turbines) into the child to a (much) higher extent than the other one. The final child will look like one of the parents with some little modifications from the other parent; thus, it will be then very likely to respect all the constraints and even compete with other feasible individuals. The result is a feasible solution. Figure 6.1a shows that the magnitude of constraint violation decreases for the entire population (until it reaches 0 m), while the parents which generate the new population are always feasible. The bigger is the population, the higher the chances are to obtain feasible children pretty soon.
By contrast, when the initialization is performed randomly (Figures 6.1b and 6.1c), the genetic algorithm immediately creates children which compete against the (infeasible) parents, but it soon stabilizes on a solution which

violates the constraints, of course to a lower extent than the initial population, but which is still unfeasible. This is not uncommon in literature. Biasing the starting population towards a good direction usually helps the algorithm be more effective; when randomly initialized, the GA spends a large amount of time just to fall in a feasible domain [73], with a higher risk of getting trapped into local infeasible minima. This problem might still be solved by considerably raising the size of the population ($\geq$ 100), but the computational effort would then become unsustainable.

The next three blocks (4, 5 and 6) combine respectively the dynamic penalty, the adaptive penalty and the repair mechanism with an *initial feasible layout*.
Regarding the first two combinations, (cases 4 and 5), a feasible starting point has been chosen because, in analogy with blocks 2 and 3, it has been found particularly hard for a genetic algorithm to start with an infeasible layout and move towards a good feasible solution with a penalty: as was clear from Figures 6.1b and 6.1c, the solution did not converge towards a feasible layout.
On the other hand, a feasible initialization, coupled with updating penalties, provided satisfactory results, because it allows the exploration of the infeasible space whilst maintaining some feasible individuals in the population, as Figure 6.2a and 6.2b show.



(a) *Dynamic penalty - Combination 4* **(b)** *Adaptive penalty - Combination 5*

Figure 6.2: *Genetic Algorithm with feasible initialization and dynamic/adaptive penalty - PAWP*

As can be seen in the figure above, for the displayed example the adaptive back-up mechanism (Fig. 6.2b), described in Paragraph 5.4.2, acts at iterations 4, 10-15: the mean LCOE raises abruptly because too many individuals are infeasible and the penalty function suddenly increases by the factor λ. At the end, the solution is feasible - the constraint violation magnitude is 0 - and *lower* (in this example just by a little quantity) *than the best individual in the starting population (i.e. the best initial guess)*. Instead, the dynamic penalty (Fig 6.2a) always increases a little bit: the constraint violation magnitude and the mean LCOE of the population are high in the first iterations, but then globally decrease.

The repair mechanism (combination n°6) obviously works with a feasible population, as the core of that block *is the feasibility*.

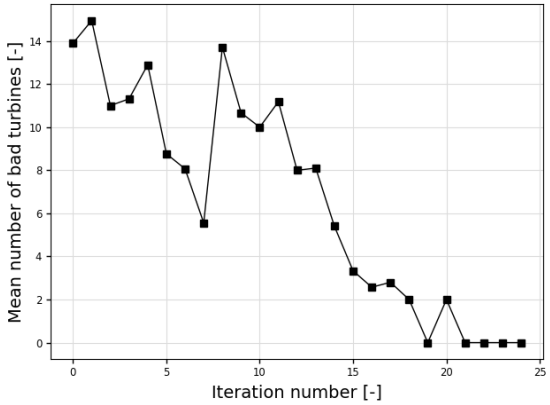


Figure 6.3: *Genetic Algorithm: mean number of turbines violating constraints - PAWP*

Figures 6.3 illustrates for the entire population how the mean number of turbines violating the constraints diminishes over the iterations. It can be clearly seen that after some iterations the algorithm starts to create only feasible individuals which do not need to be repaired anymore.

The 7_{th}, 8_{th} and 9_{th} block just evaluate the grid-based approach. This is characterized by feasible generations. It is just important to note that if two parents generate a child which has two identical genes - if one layout has two turbines in the same grid point ($[x, y]$ position) - a "repair" mechanism has been implemented: one of the two genes is randomly replaced by another point belonging to the grid.

Particle Swarm Optimizations

Similarly to the first three blocks, combinations n° 10, 11 and 12 concentrate on the way static penalties deal with different initializations. Differently from the GA, none of these combinations was able to provide a final feasible solution. If the PSO is initialized with a feasible population - combination n° 12 - no progress is done by the optimization because all the new particles cannot compete with the (good) initial guess. Instead, when the initialization is randomly performed, the algorithm gets stuck into local (infeasible) minima. This can be noticed from the flat trend towards the last iterations in Figure 6.4: all the new solutions computed in each iteration cannot compete anymore with the previous ones and the algorithm sticks to the same value.

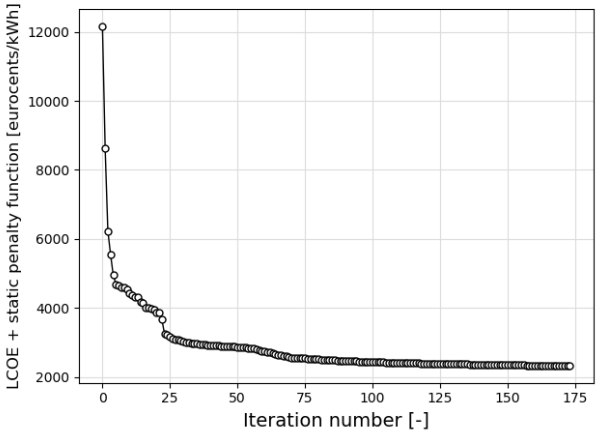


Figure 6.4: *PSO: Value of the best global position of the swarm - static penalty function and random initialization (EL) - combination 10*

Figure 6.4 shows the pattern of the global best position of the swarm. Although some significant improvements w.r.t. to the first iterations takes place, this is not enough to ensure a fully feasible solution[1]). Multiple combinations of the inertia factors C_0, C_1 and C_2 (see Equation 4.1)(used for the calculation of the speed of one particle) have been tried, without success. The reason may lie in the fact that a random initialization does not allow some more uniform placement of the turbines. A possible solution might be the exploitation of *quasi-random* techniques which decrease the overall starting infeasibility of the swarm [74] (without making the population totally feasible), but this possibility has not been explored in this work.

Combinations n° 13 and 14 analyse how the dynamic and the adaptive penalty functions deal with feasible initializations. Interestingly, *none* of the analysed combinations provided satisfactory results. In particular, *none of these blocks was able to make some progresses.* This means that after the first 1-2 iterations, the particles do not update anymore, as no better solutions are found over the course of the optimization. The explanation is as follows. In case of a feasible initialization, the inertia factors C_0, C_1 and C_2 create, during the first iterations, highly infeasible individuals which are not able to compete with the initial guess, even though the dynamic and the adaptive penalty functions allow a high exploration of the design space (the C factor is very low, as explained in Chapter 5[2]). Therefore, there are two ways to solve this issue: the first is to reduce the inertia factors and the second is to further reduce the C factor in the penalty function definition. But, if the former is applied, then the PSO would lose its capacity to explore the design space: too similar particles would be created from one generation to another, inhibiting the swarm from finding a solution which is not highly dependent on the initial guess (too pronounced similarity); on the other hand, if the latter approach is applied, the particles update over the iterations, as now the penalties are so low that the competition among old and new

[1] the extreme high LCOE values show that a penalty is still applied. Thereofore, the solution is still infeasible

[2] The C factor *is not* one of the inertia factors! It is defined in the penalty mechanism (Equations 5.4 and 5.6).

individuals is enhanced, however this phenomenon lasts just for some more iterations (around 5-6) and no more progress is achieved. The actual root of the problem has not been identified yet. It seems particularly hard for a PSO not to include too highly infeasible individuals in the optimization. That is maybe the reason why there are no examples of dynamic or adaptive penalties applied to this algorithm. Some authors use techniques such as a *multi-swarm* PSO [75] or a distance-based fuzzy function to determine the penalty coefficients [76], however *these approaches are far from the penalties described in the previous chapter*. Since no further explanation can be ultimately given, these approaches have not been considered anymore.

Combination n°15 deals with the repair mechanism coupled with the feasible initialization. Figure 6.5 shows that the mean number of violating turbines diminishes over the iterations, however, differently from the GA (and, as shown in a few lines, the DEA), this number does not approach to zero towards the end of the optimization. The reason lies in the fact that on the one hand the GA and the DEA are *evolutionary-based*, meaning that the entire population *converges* towards the minimum as it is part of the evolutionary process; on the other hand, in the PSO the particles are much more *independent* from each other. Therefore, when the stopping criteria is satisfied and the optimization stops, there is no overall *similarity* between the swarm particles and thus no convergence towards zero-violating turbines.

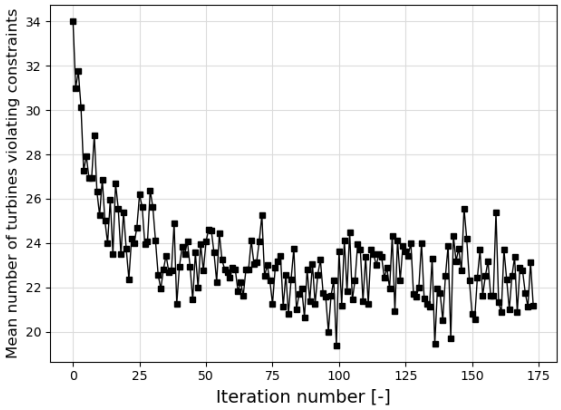
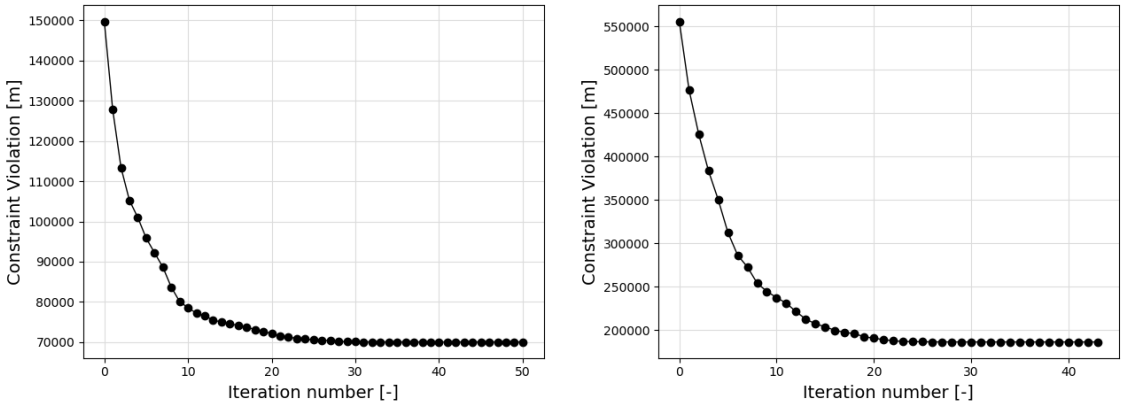


Figure 6.5: *PSO: mean n° of turbines violating constraints, repair mechanism, feasible initialization - EL*

Differential Evolutionary Algorithms

As mentioned slightly above, the 16_{th} and 17_{th} block consist of a *hybrid initialization* - i.e. two initializing mechanisms are chosen to create one population - and a static penalty. This has been done due to the issues which combination n° 18 - same as 16 and 17 but with *one* initializing technique only - creates on the overall convergence. In other words, block n° 18 starts from an infeasible population. When combining the individuals to create a new one for the next iteration (see Equation 4.3), the static penalty approach makes the optimization unable to escape from local (infeasible) optima, in the same way as for combinations n° 2, 3, 10 and 11 (Figures 6.6a and 6.6b).



(a) *Random modified initialization* **(b)** *Random initialization*

Figure 6.6: *The "DEA-static-random initialization": population constraint violation plot. For reasons of simplicity, the population is initialised first via randomly and then via random-modified initialization, despite belonging to the same combination (Combination 18)*

The DEA's mutation-based way of creating new individuals in the population decreases the magnitude of the constraint violations, as can be seen in Figures 6.6a and 6.6b but, since the individuals are distributed randomly, it is extremely difficult to eventually bias the search towards a feasible solution: there is not enough uniformity in the starting turbines' placement, hence the DEA is not able to force *all* the turbines to either stay within the boundaries or to be at least $4D$ far from each other. Some authors claim that higher probabilities to obtain feasible solutions may be yielded if the penalty was further increased: this typically results in a fast convergence to a feasible result [43]. However, in this work the difference between the LCOE of a feasible solution (order of magnitude: $10^1€/kWh$) and of a penalized solution (order of magnitude: more than $10^3 - 10^4€/kWh$) is already so big that further increasing the penalty would not lead to significant improvements.

On the other hand, combo n° 19 (static penalty with feasible initialization) is completely useless as it does not allow the iterations to progress. In fact, since the DEA creates new individuals by adding/subtracting their $[x, y]$ turbine coordinates, it is evident that all the new individuals, from the second iteration onwards, will be *extremely* likely to violate the constraints. When this happens, the static penalty increases the objective function and none of the new individuals is able to "win" against the starting feasible ones.

That is why the hybrid initializations were done. Since with the static penalty approach the results from either a totally infesible or feasible population were not good, the idea of combining them arose. If half of the initial population is feasible and the other half is infeasible, then during the first iterations some new individuals might actually compete against the *infeasible* half. Interestingly, *after some time the optimization is able to find even better individuals than the initial feasible ones.*
The reason for that is shown in the following picture.

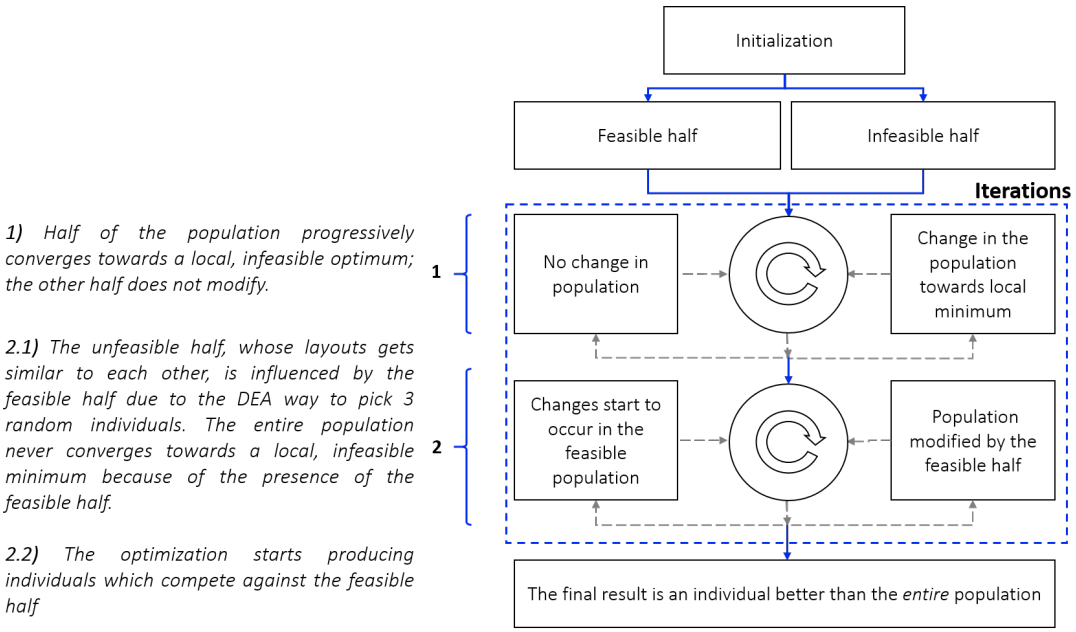
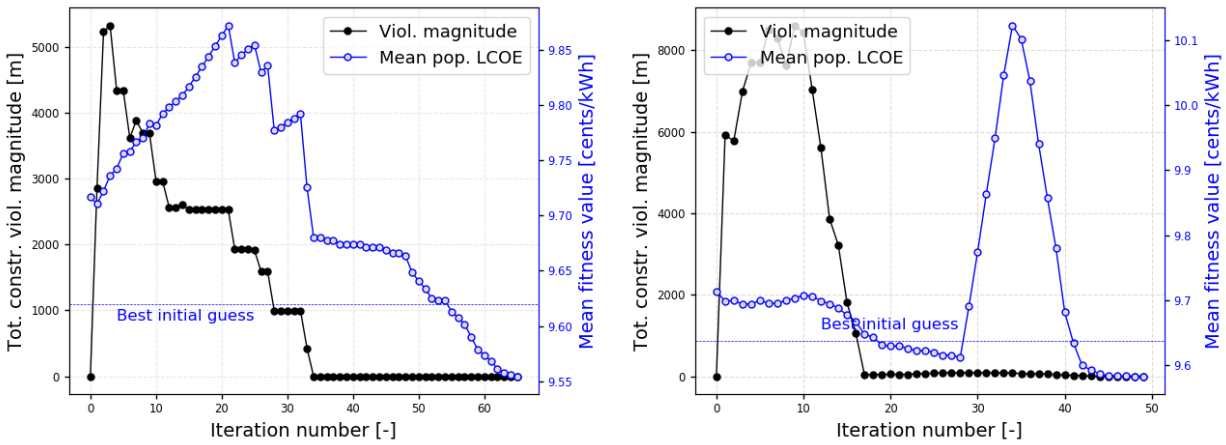


Figure 6.7: *The "DEA-static-hybrid initialization" optimization*

In the first phase[3], only the infeasible half is modified over the iterations. The mutation mechanism in Equation 4.3 keeps randomly picking three individuals from both halves. However, when the infeasible half moves towards a certain local (infeasible) minimum - as happens in block n° 18 - the individuals belonging to that half will start looking like each other. Therefore, *there will be the growing probability that, when Equation 4.3 picks three random individuals, similar individuals get picked and maybe slightly edited by the feasible individuals.* An example could be: after some time, layouts i and j come from the infeasible half and are similar to each other; layout k comes from the feasible part. When combined by Equation 4.3, the resulting individual will look like layouts i and j *with some "feasible" features coming from* k. After some time, this will make the population move towards feasible solutions, as the final output will be a layout with "infeasible origins" which has been continously modified by the feasible half.

Cases 20 and 21 respectively deal with the combination of the DEA-feasible initialization and dynamic/adaptive constraint handling technique. The plot of the mean LCOE and the constraint violation magnitude is shown below.

[3]Note: there are no "phases": this distinction was done just to enhance the comprehension of the hybrid initialization



(a) *Dynamic penaly* **(b)** *Adaptive penalty*

Figure 6.8: *Differential Evolutionary Algorithm with feasible initialization and dynamic/adaptive penalty - PAWP*

In analogy with Figures 6.2a and 6.2b, the adaptation mechanism (Fig. 6.8b) suddendly increases the penalty (at iteration 28); here, at iteration 27, the magnitude of the constraint violation is so low that even if the iteration is on a late stage, the infeasible individuals compete against the feasible. Therefore, λ raises until the constraint violation magnitude (in black) eventually goes to 0; the dynamic penalty plot (Fig. 6.8a) shows the same pattern as for the genetic algorithm.

In analogy with the GA, when the repair mechanism (coupled with a feasible initialization - block n°22) is applied to the DEA, for the entire population the mean number of turbines which violate the constraints decreases as long as the optimization runs.

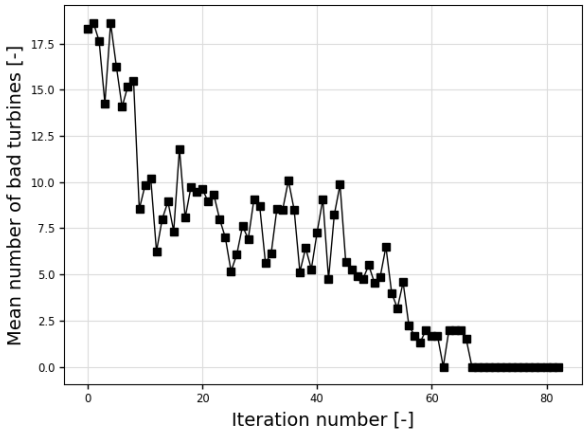


Figure 6.9: *Differential Evolutionary Algorithm: mean number of turbines violating constraints - PAWP*

As can be noticed, even in the DEA the population, after some iterations, stabilizes on generating only feasible layouts, even if it takes longer than in the GA.

6.3 Assessment of the combinations

This section explains the assessment criteria which have been used to score the different combinations. After that, the results and the consequent ranking are shown.

6.3.1 Presentation of the assessment criteria

The criteria and metrics used to evaluate *each block* are either taken from literature [9] or defined by the author. These are here itemized. Since the final goal is to perform a multi-criteria analysis of the blocks, *all the results from the assessment criteria are normalized from 0 to 1. The smaller the scores are, the better performing one block is.*

- **Optimality** [9] ($M_{optimality}$).
  This criterion defines the absolute goodness of the final result from the optimization (in this case, *how low the LCOE is*):
  $$M_{optimality} = LCOE \tag{6.1}$$

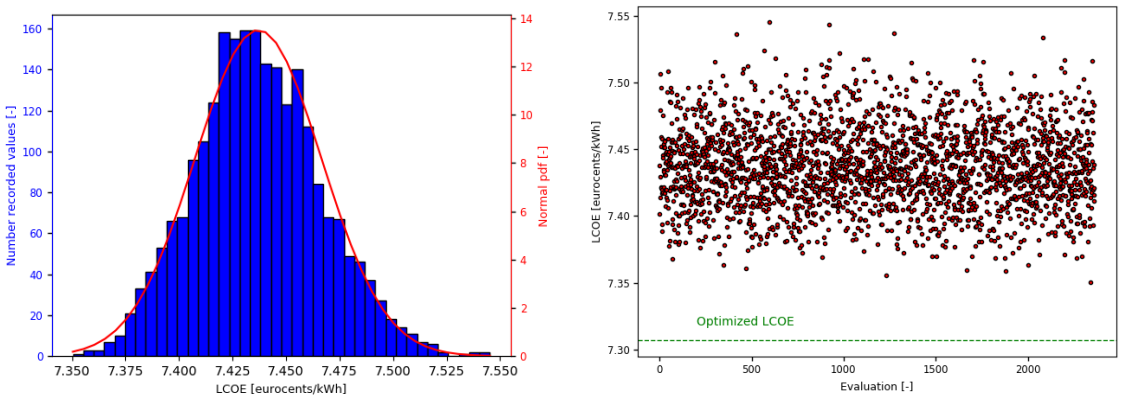
- **Feasibility** [9] ($M_{feasibility}$)
  Since some blocks may lead to *slightly* infeasible layouts (overall small constraint violation) which can be further processed and modified by the user, the extent of this (low) violation is measured by this criterion. This is done by calculating the final overall magnitude of the constraint violation:
  $$M_{feasibility} = \sum_{i=1}^{N_{turbines}} (g_1(\mathbf{x}) + h_1(\mathbf{x})) \tag{6.2}$$

- **Precision** [9] ($M_{precision}$).
  The user is obviously interested in a block which gives the same optimum *consistently*. $M_{precision}$ measures the *standard deviation* of the optimal solutions yielded by multiple runs of the same combination.
  $$M_{precision} = \sqrt{\frac{\sum_{i=1}^{N_{runs}} (LCOE_i - LCOE_{mean})^2}{N_{runs} - 1}} \tag{6.3}$$

- **Speed** (M_{speed}).
  This criterion assesses the speed of the selected block. Since the time for one iteration might change for each algorithm, as well as the mean number of required iterations to find a solution, M_{speed} is evaluated by taking the product of the mean time to complete iteration and the mean number of iterations:
  $$M_{speed} = n_{iterations_{mean}} \cdot T_{iteration} \tag{6.4}$$

- **Random sampling** ($M_{random-sampling}$).
  Each optimization gets an output after a certain number of *iterations*; in every iteration, the *objective function* (i.e. the LCOE) of every individual in the population (p) is evaluated. This means that the total number of objective function evaluations in one run is equal to $N_{evaluations} = n_{iterations} \cdot p_{size}$. The random sampling criteria consists of calculating the LCOE of a population of layouts [4], which is *equal* to $N_{evaluations}$.
  Since different initializing techniques are available to create layouts, *that population is initialized in the same way as the block under consideration.* The LCOE of all the individuals in the population is computed in WINDOW and the distribution of the results is drawn. It has been observed that the data follow a *normal* trend. The cumulative distribution function is then calculated. This is the expression written in Equation 5.7. The mean value of the objective function from the optimization is then compared to the set of random results. If the former is *lower* - i.e. better - than the smallest value from the set, this means that with the same $N_{evaluations}$, the optimizing block being analyzed performs *better* than a pure random search; by contrast, if some LCOE's from the set are lower than the LCOE from the optimization, then the optimizing block falls into local minima and therefore performs *worse* than a pure random search.
  However, since random search is determined by pure luck, there is no certainty that if one run is repeated, the resulting set will still yield a better LCOE w.r.t. the optimized one. Therefore, a *statistic* approach is used: the cumulative distribution function which is yielded after one run gives the probability the random search has to provide a better LCOE than the one computed by the optimizing block. *The closer this probability is to 0, the better-performing the optimizing block is.* The random sampling criterion can eventually be defined as follows:
  $$M_{random-sampling} = \frac{1}{2}[1 + erf(\frac{LCOE_{optimized} - \mu_{rand-search}}{\sigma_{rand-search}\sqrt{2}})] \qquad [-] \tag{6.5}$$
  being μ and σ the mean and the standard deviation from the LCOE distribution computed in the random search. An example is provided below. The Particle Swarm Algorithm coupled with a repair mechanism for Eneco Luchterduinen is displayed. As can be seen, the random search provides LCOE values which follow quite well a normal distribution (this happens for all the other optimizing blocks - see Appendix D). In the case displayed in Figure 6.10a and 6.10b, the optimizer is *better* than the random search. The probability for the random search to find a better LCOE than the optimization is *extremely* low.

[4] This means that only WINDOW is used and *no optimization is carried out.*



(a) *Fitted normal distribution of random sampled data* **(b)** *Scatter random sampled data*

Figure 6.10: *Random sampling: histogram and scatter plot - example from Eneco Luchterduinen*

- **Robustness** ($M_{robustness}$).
  The robustness evaluates the capacity of a certain block to perform similarly for each case study. It is the only criterion which depends on the others and this is why it has to be chronologically analyzed *last*. All the values from the other criteria are *normalized*. In this work, the robustness has been defined as the *standard deviation of the normalized scores - from the other assessment criteria - in each use case*:

$$M_{robustness} = \sum_{j=1}^{N_{criteria}=5} \left(\sqrt{\frac{\sum_{i=1}^{N} (score_i - score_{mean})^2}{N-1}} \right)_j \quad \forall \quad assessment \quad criteria \tag{6.6}$$

  N is the number of case studies (in this work 2, i.e. EL and PAWP); $score_i$ is the score that one assessment criterion yields w.r.t to case study i and $N_{criteria}$ is the number of assessment criteria (here: 5).

6.3.2 Overview of the scores

The final scores for the two case studies are displayed in Tables 6.2 and 6.3. Since several identical optimizing runs have been done for each block, the final scores are *the mean* of the results. Data must be read with the rule "the lower, the better".

Table 6.2: *Overview of the scores for PAWP case study*

Case study 1 - Prinses Amalia WindPark						
Block		**Optimality** [c€.kWh^{-1}]	**Feasibility** [m]	**Precision** [-]	**Speed** [s]	**Random sampling** [-]
1	GA-static-feas.	9.55	0	0.0369	1735.98	0.00768
2	GA-static-rand.	9.68	109.486	0.0251	1351.46	0.04682
3	GA-static-rand. mod.	9.66	973.6666	0.0288	1542.05	1.17E-02
4	GA-dynamic-feas.	9.54	0	0.0535	876.5	2.12E-03
5	GA-adaptive-feas.	9.57	0	0.0278	1116.42	0.01858
6	GA-repair-feas.	9.44	0	0.0317	1389.59	2.41169E-06
7	GA-equil. triangle	9.56	0	0.0776	1302.76	2.416E-08
8	GA-rand. triangle	9.52	0	0.0599	1140.45	0.0199
9	GA-square	9.76	0	0.4462	2145.22	2.90E-14
10	PSO-static-rand.	9.77	420.03	0.0148	1529.045	0.3148
11	PSO-static-rand.mod.	9.77	312.5	0.063498189	1564.70	0.3148
12	PSO-static-feas.	no progress				
13	PSO-dynamic-feas.	no progress				
14	PSO-adaptive-feas.	no progress				
15	PSO-repair-feas.	9.44	0	0.0229	8640	2.85E-06
16	DEA-static-rand.+ feas.	9.54	0	0.0397	5210.28	0.0071
17	DEA-static-rand. mod. + feas.	9.57	0	0.0309	4499.28	0.02748
18	DEA-static-rand.	9.71	2210.4	0.1085	3940.41	0.10836
19	DEA-static-feas.	no progress				
20	DEA-dynamic-feas.	9.53	0	0.0379	3599.94	0.00159
21	DEA-adaptive-feas.	9.56	0	0.0418	3868.37	0.00579
22	DEA-repair-feas.	9.42	0	0.0354	4406.96	7.90E-07

Table 6.3: *Overview of the scores for EL case study*

Case study 2 - Eneco Luchterduinen						
Blocks		**Optimality** [c€.kWh^{-1}]	**Feasibility [m]**	**Precision [-]**	**Speed [s]**	**Random sampling [-]**
1	GA-static-feas.	7.36	0	0.016	739.64	0.00871
2	GA-static-rand.	7.413	1024.6	0.0542	2834.04	0.1065
3	GA-static-rand. mod.	7.440	1092.3	0.0127	964.593	0.005727334
4	GA-dynamic-feas.	7.35	0	0.0299	791.375	0.0016
5	GA-adaptive-feas.	7.364	0	0.03047	614.631	0.00323
6	GA-repair-feas.	7.32	0	0.0184	1298.192	6.07E-05
7	GA-equil. triangle	7.43	0	0.0048	665.43	7.22E-05
8	GA-rand. triangle	7.3388	0	0.0214	622.568	6.50E-07
9	GA-square	7.9551	0	0.0327	872.48	0.00033
10	PSO-static-rand.	7.5233	1167.6	0.0461	1142.05	0.9169
11	PSO-static-rand.mod.	7.48	788.6	0.0301	787.05	0.0301
12	PSO-static-feas.	no progress				
13	PSO-dynamic-feas.	no progress				
14	PSO-adaptive-feas.	no progress				
15	PSO-repair-feas.	7.33	0	0.00158	25072.05	0.0002
16	DEA-static-rand.+ feas.	7.365	0	0.0136	3255.195	0.0124
17	DEA-static-rand. mod. + feas.	7.39	0	0.0226	2924.324	0.0906
18	DEA-static-rand.	7.478	5757	0.0345	3134.175	0.6177
19	DEA-static-feas.	no progress				
20	DEA-dynamic-feas.	7.379	0	0.0092	1745.975	0.0237
21	DEA-adaptive-feas.	7.38	0.06	0.0159	1968.037	0.05491
22	DEA-repair-feas.	7.31	0	0.0106	6596.887	5.22E-06

As mentioned in Paragraph 5.3.3, the triangled-random grid, due to its irregular shape, obtains a better optimality than the other grid-based approaches. As can be further seen from above, the Robustness has not been reported in the tables. Since that criterion depends on the other criteria and on the number of case studies, there are no values for individual cases. The way Robustness has been used is described in the next paragraph.

6.3.3 Selection of the best optimizing blocks - Multi Criteria Analysis (MCA)

In order to fairly determine the best optimizing block, a multi-criteria analysis (MCA) has been performed. By definition, a MCA consists of a set of m alternatives $(A_1, A_2, ..., A_m)$ and n decision-attributes criteria (also called *benefit attributes*) $(R_1, R_2, ..., R_n)$ [77]. Every decision criterion is assigned a pre-defined *weight* ω_i $(i \in [1, n])$ which establishes the relative importance of one criterion over the others. In this thesis, the alternatives are the *optimizing blocks* and the decision-attributes are the *Optimality, Feasibility, Precision, Speed, Random Sampling* and *Robustness*.
The preliminary step to carry out a correct MCA is to normalize the original data [78]. After that, several methodologies can be used in the decision-making process. In this paper, two of the most popular techniques have been chosen: a Simple Additive Weighting method (SAW) and a Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [77] [79].
It is important to stress that some extensive research has been done on the influence of the *data normalization* on the multi-criteria decision-making technique. Vafaei et al. [79] [77] generally acknowledge, thanks to considerations based on the calculated average result deviation for each normalization technique, the best normalizing method to be the *vector normalization*. In other words, the score that one alternative has based on a certain benefit attribute (shown in Tables 6.2 and 6.3), is normalized as follows [79]:

$$r_{ij} = 1 - \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \tag{6.7}$$

The 1 in Equation 6.7 is used because in Tables 6.2 and 6.3 the best values are the lowest ones, whereas a multi-criteria analyis is usually performed by *maximizing* the score that an alternative gets. Equation 6.7 is used both for TOPSIS and SAW, however some authors [80] also suggest using a linear scale normalization (which is more intuitive) to be implemented into the SAW approach:

$$r_{ij} = 1 - \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}} \tag{6.8}$$

Both these combinations have been tried. The values from Tables 6.2 and 6.3 have been normalized both via Equation 6.7 and by Equation 6.8. After that, a single table made of the mean values is built. The Robustness

uses the data from both Table 6.2 and Table 6.3 to yield a single value for each block. This value is normalized and then it simply becomes a column in the mean-values table. To enhance the comprehension, the table below shows the vector-normalized mean values from the combination of the two previous tables.

Table 6.4: *Overview of the mean vector-normalized scores for the two case studies (PAWP and EL): high values indicate a better performance. In the original table, numbers are rounded up to the 6 decimals*

Block	Optimality [-]	Feasibility [-]	Precision [-]	Speed [-]	Robustness [-]	Random sampling [-]
1	0.766	1.00	0.888	0.926	0.852	0.988
2	0.763	0.89	0.732	0.901	0.622	0.902
3	0.763	0.71	0.913	0.929	0.774	0.985
4	0.766	1.00	0.811	0.955	0.835	0.997
5	0.765	1.00	0.835	0.950	0.788	0.978
6	0.768	1.00	0.885	0.928	0.909	1.000
7	0.764	1.00	0.899	0.943	0.913	1.000
8	0.766	1.00	0.843	0.949	0.860	0.978
9	0.754	1.00	0.398	0.909	0.859	1.000
10	0.761	0.82	0.778	0.926	0.517	0.249
11	0.761	0.87	0.800	0.931	0.660	0.646
16	0.768	1.00	0.969	0.235	0.991	1.000
17	0.766	1.00	0.898	0.759	0.807	0.987
18	0.765	1.00	0.867	0.790	0.759	0.930
19	0.762	0.08	0.735	0.805	0.612	0.606
21	0.766	1.00	0.920	0.843	0.861	0.988
22	0.765	1.00	0.886	0.829	0.810	0.969
23	0.768	1.00	0.916	0.725	0.921	1.000

The next step is to choose appropriate values for the weights ω_i. These will multiply each cell of the table above. Several ways to build well performing weights exist. In multi-criteria decision analysis, they are usually classified into three categories [78]: *subjective* (defined by the user), *objective* (defined by some algorithms) and *combined*. In this work, the former has been chosen, due to its simplicity and flexibility. The weights are listed in Table 6.5. It is crucial to stress that the sum of ω_i *must* be 1.

Table 6.5: *List of weights ω_i applied to the assessment criteria C_i*

Optimality	Feasibility	Precision	Speed	Robustness	Random Sampling
0.25	0.1	0.1	0.1	0.2	0.25

The *optimality* and the *random sampling* have the highest influence: the former because of the desire to lower the LCOE of an OWF; the latter because this criterion evaluates the goodness of one optimizing technique w.r.t. to pure random search (see Paragraph 6.3.1). On the other hand, the *robustness* is also meant to play an important role as it appraises the performance of one block over multiple case studies. The remaining attributes have the same relative importance.

After having normalized and weighted all the values, either the SAW or the TOPSIS approach can be applied. The former consists of simply adding each element belonging to the same *row*. The final result is the overall score of each optimizing block; the latter is slightly more complicated and is described in full detail in Appendix B. The final ranking of the combinations listed in Table 6.1 are shown below.

Table 6.6: *Combination ranking for different multi-criteria analyses*

TOPSIS vector-normalized	SAW vector-normalized	SAW linear-sum normalized
1) GA-equilateral triangle	1) GA-equilateral triangle	1) GA-equilateral triangle
2) GA-repair mech.-feasible	2) GA-repair mech.-feasible	2) GA-repair mech.-feasible
3) DEA-repair mech.-feasible	3) DEA-repair mech.-feasible	3) GA- static pen.-feasible
4) GA-static pen.-feasible	4) GA-static pen.-feasible	4) DEA-repair mech.-feasible

The three multi-criteria analyses give almost the same results. There is some little misalignment at the third and fourth place, but since literature generally identifies the vector normalization to be the best in both analyses (TOPSIS and SAW), the third place is assigned to the "DEA-repair mech.-feas" block.
A few remarks have to be stressed.

- The repair mechanisms have the best results in terms of optimality and random sampling. In particular, the Differential Evolutionary Algorithm always yields a lower LCOE w.r.t. to the Genetic Algorithm in all the analyzed cases, whilst the random sampling score is similar. The reason is because the GA exchanges information thanks to the crossover and therefore it works with combinations of a *limited* set of values, i.e. the initial population. Although there is mutation, which is meant to prevent GAs from falling into local minima, this may not be sufficient to overcome the optimality of the DEA mutation-based way to create new individuals. In other words, the DEA has more freedom than the GA when generating a population.
- The Genetic Algorithm is, on the other hand, much faster than the DEA and therefore is recommended to those users who are *slightly* less interested in performing a detailed design and prefer having something quicker with little loss of optimality. The higher speed of the GA can still be justified due to the mechanism of producing new individuals over the iterations. Due to their combinatorial nature, Genetic Algorithms converge faster as they deal with a "closed group" of variables.
- The outputs from the repair mechanisms in different case studies are also consistent with each other (good robustness) and precise. The spread of the LCOE values in identical runs is low, as well as the deviation of the scores for different use cases.
- The equilateral-triangle initialization is characterized by very good values for optimality and random sampling and excellent results in terms of precision and robustness. However, as explained in Paragraph 6.3.1, when the random sampling assessment criterion is applied to a block, the LCOE of $N_{evaluations}$ layouts is performed by using the *same* initializing technique of *that* block. Therefore, the excellent "random sampling value" which the combination "GA-equilateral triangle" yields is because of the starting layouts which are triangle based and not - as many other combinations - created by means of random feasible initialization (which usually provides lower LCOEs).
  Triangle-based optimizations are precise, robust, produce feasible solutions and have some sufficiently good optimality, although a random search (performed with random *feasible* individuals) would obtain lower LCOEs. However, since triangle-based procedures are also fast - much faster than pure random search with feasible initializations - they are suitable for preliminary estimations, which do not require much accuracy but at least give the designer some guidelines on what to expect from a certain OWF.

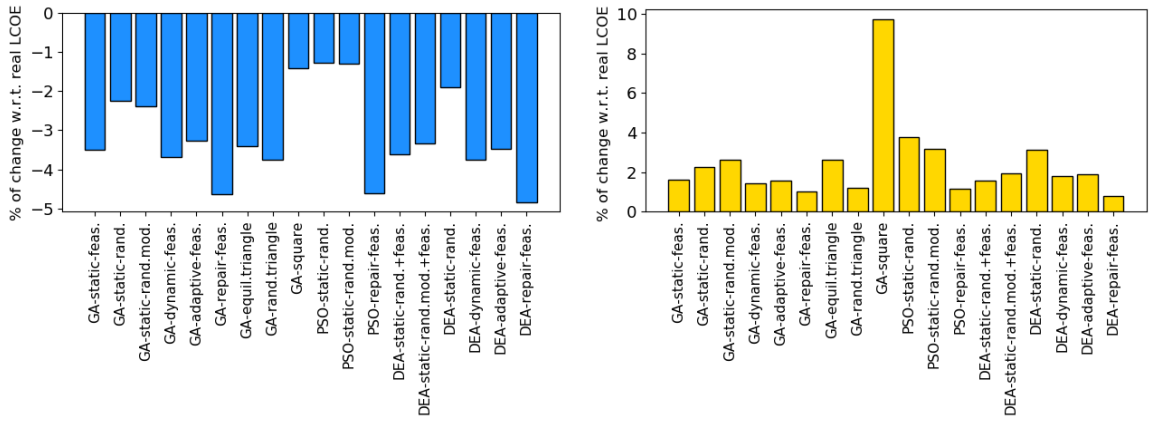
Based on these considerations, *the optimizing procedures which the author suggests using are the repair mechanism-based DEA and GA.*

6.4 Comparison with the real wind farms

In this section, the results from the optimizations are compared to the realised LCOE of the case studies. The results are described and commented in Paragraph 6.4.1. Since some important issues rose on the influence of the wind direction sampling on the optimizations, Paragraph 6.4.2 analyzes this in detail.

6.4.1 Overview of the results

In this paragraph, the mean value of the LCOE from each optimizing technique is shown for Prinses Amalia and Eneco Luchterduinen. Figures 6.11a and 6.11b illustrate the deviation in percentage of the new LCOEs from the *realised* levelised cost of energy of the wind farms[5].



(a) *PAWP* **(b)** *EL*

Figure 6.11: *Mean percentage of change from real to optimized LCOE*

[5]The real layout of the wind farm has been set in WINDOW and the LCOE has been calculated

As can be noticed from the bar charts, for PAWP *all* the combinations yield a better LCOE than the real case - and the best results are achieved by the repair mechanisms - but, interestingly, no optimizing block is able to produce a result which is better than the real case (the change in percentage is positive) for Eneco Luchterduinen. Three important observations need to be stressed.

- First of all, Figures 6.11a and 6.11b provide clear evidence that the best LCOE values are achieved by the repair mechanisms. In fact, these yield the highest negative percentage deviation for PAWP and the lowest positive deviation for EL. This is not a surprise, as these are the blocks with the best *optimality* score.
- All the written optimizing codes are meant to provide *irregular* layouts, whereas, as shown in Figures 3.3 and 3.5, both the real layouts of PAWP and EL are *regular*. The outcome from the driver is always an irregular layout both because of the *stochastic elements* which characterize the algorithms and the fact that the *initial population*, even in grid-based approaches, *is never regular* (so that it is difficult to bias the algorithms towards a regularly shaped layout). It is crucial to note that even if the LCOEs from the simulations are *worse* than the real case for EL, in all the runs the optimizers *always found a better solution than the layouts in the initial (irregular) population* (and the best layouts yield an LCOE which is 1% more than the baseline). Further evidence of the good performance of the algorithms was obtained as follows: the *real layout* of EL (with some little modifications) was *used as one of the individuals* in the population. In that case, the optimal solution was *better than the actual layout*. Thus, if fed with a (good) regular layout, the algorithm obtains a better value than the baseline. Indeed, no accurate study has been carried out in this thesis about the goodness of a regular layout w.r.t. an irregular one for the optimization. It is important to note that the wind farm boundaries of Eneco Luchterduinen are even much tighter than in Prinses Amalia. It can be consequently possible, for a regular layout, to be better performing in a tight area as some better exploitation of the wind resource can be achieved. Literature usually agrees upon the fact that irregular-shaped layouts perform generally better than regular ones [7][37][70], but this is not a rule. Therefore, further research is advised in order to better investigate on the effect of a regular population as a starting point for the optimization.
- The wind direction sampling used to calculate the wake effects in each sector was set equal to 45° (Paragraph 3.3.3). Therefore, it would be interesting to investigate on the influence of the wind direction sampling stepsize to see whether the regular layouts might really perform better than irregular ones. In fact, eight wind direction sectors may not be enough to properly assess how the wind farm captures the wind resource. This is investigated in detail in the next paragraph.

6.4.2 The influence of the wind direction sampling stepsize

The reason why the real (regular) layout of Eneco Luchterduinen performs better than the ones from the optimizations may be due to an *artificial* improvement, i.e. the cause may lie in the relatively rough wind direction sampling which has been used (8 sectors of $\theta = 45°$ each, being θ the angle drawn by each sector). As written by Porté Agel et al. [81], using different wind direction sampling angles can be interpreted as *changing the layout of the wind farm* relatively to the incoming wind, whilst maintaining the same turbine density. An article about the correlation between wind sampling and optimization was written by Feng and Shen in 2015 [2]. In that paper, the authors analyse how modelling the wind direction distribution may have an impact on the overall optimization of the wind farm layout. To let the reader acknowledge how tricky this topic may be, they provide an example. A fictitious wind farm is built. This is made of three turbines only, which are located at the edge of a circle and which are equally spaced (blue circles). A second, identical wind farm is built in the same way, but 15°-rotated (in red), as Figure 6.12 shows.

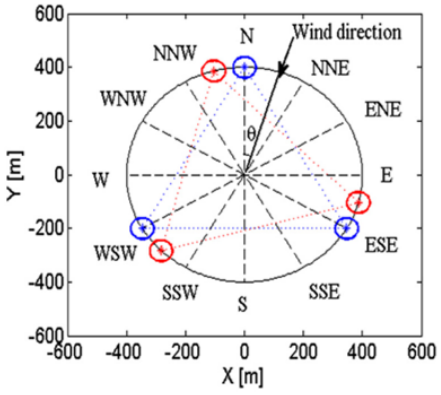


Figure 6.12: *Fictitious identical wind farms. The one in red is slightly rotated by 15° [2]*

If this example was under the assumption of a constant wind speed, *uniformly* distributed among *all* the directions (same probability for each angle), it could be concluded that Layout 1 (in blue) and Layout 2 would have the same power output. However, if the wind direction sampling stepsize was equal to 30 degrees, the wind probability was still the same in each sector and the wake zones were plotted[6], the power output would be different for the two wind farms due to the wake effects, as next figure shows.

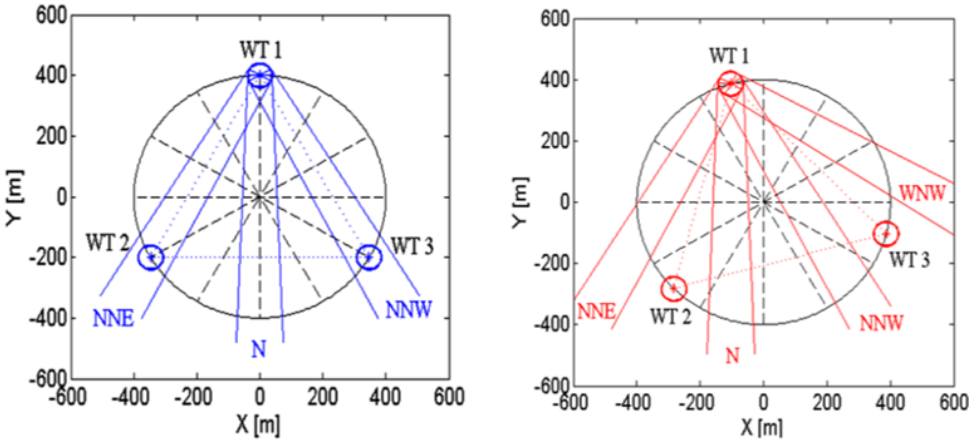


Figure 6.13: *Wake development in the two layouts being $\Delta\theta = 30°$. The red one performs better than the blue one [2]*

As Figure 6.13 shows, the expected power output from Layout 2 would be higher than Layout 1. This contradicts the previous observation, which stated that the power output should be the same. This means that if the wind is discretized into 12 sectors ($\theta = 30°$), one alternative becomes better than another. Nevertheless, in reality wind comes from all the directions so that this improvement might actually be artificial. Something similar is therefore likely to happen even for the case studies in this thesis. In other words, the real (regular) layout - in this case of Eneco Luchterduinen - could perform *worse* than the ones from the optimization, if more than 8 sectors were considered.
Therefore, the effect of the step size for the angle discretization has been studied.

Wind direction sampling without optimization

In Figure 6.14, the LCOE of a few *random irregular* layouts is compared to to the actual LCOE of Eneco Luchterduinen (these layouts *are not optimized*). For each layout, the levelised cost of energy has been evaluated in WINDOW for different step sizes.

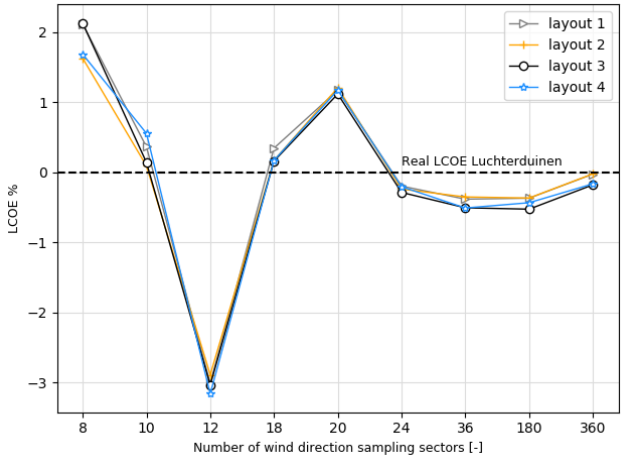


Figure 6.14: *LCOE difference w.r.t. EL actual layout - re-evaluated for different stepsizes for 4 random layouts*

The reason why the trends are so close to each other lies in the fact that a different wind direction sampling affects only the *aerodynamic performance* of the wind farm. The energy production affects the LCOE, but the

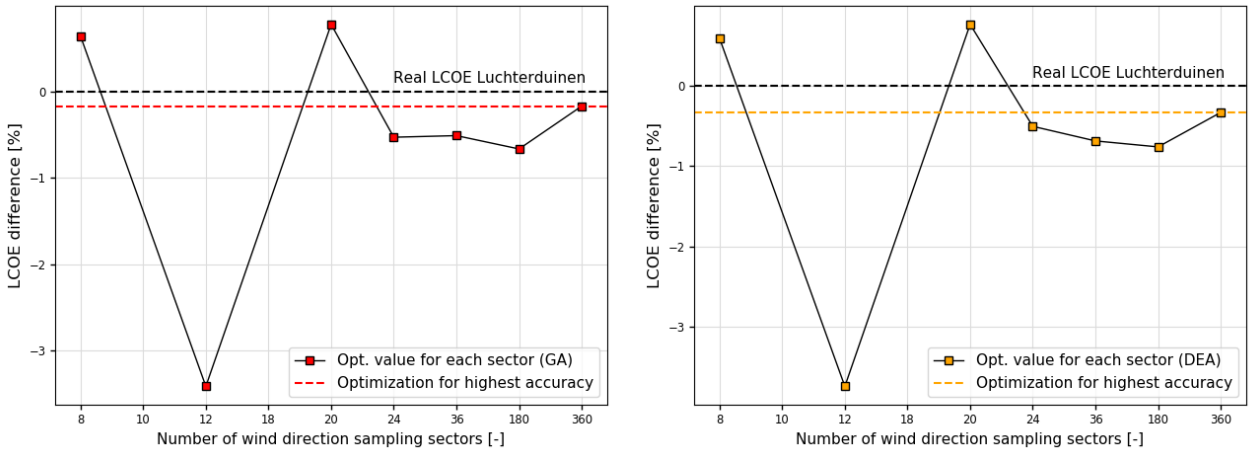
[6] the authors use the Jensen model and assume the wake decay coefficient $\alpha = 0.04$

decrease (or the increase, depending on the chosen sampling) is overall small, resulting in a little variation in the final LCOE value.

The figure shows a particular pattern. When the sampling is relatively rough (from 8 to 20 sectors), the four random layouts obtain a lower LCOE than the one from the real, regular layout of Eneco Luchterduinen if evaluated over 12 sectors, whereas they yield higher LCOEs than the real design when evaluated over 8 (the set default value), 10, 18 and 20 sectors; by contrast, when the number of wind direction sampling sectors is 24 or more, the LCOE difference w.r.t. to the real layout converges towards a stabler value (the trends are significantly smoothened) and a higher *accuracy* is yielded.
Indipendently from the actual LCOE values, i.e. if they are better or worse than the actual LCOE[7], these observations clearly show how a relatively *rough wind direction sampling* may yield *contradictory results*, in compliance with the observations of Feng and Shen [2]. Furthermore, some evidence of how the LCOE improvement might be artificial in some cases (12 sectors) is given.

Wind direction sampling with optimization

If the layout *optimization* is carried out *each time over a different number of sectors*, both by means of GA and DEA, the outcome is as displayed in Figures 6.15a and 6.15b.



(a) *GA* **(b)** *DEA*

Figure 6.15: *LCOE improvement (EL) with optimization carried out each time with a different stepsize*

From these figures the following observations can be drawn.

- All the depicted values from the GA and the DEA are *lower* than those in Figure 6.14. This is because the results in Figure 6.15a and 6.15b come from a sector-based *optimization*, which obtains better (i.e. lower) values than the four *random* layouts in Figure 6.14 (which were not optimized).
- It can be further observed that the trends are in compliance with those depicted in Figure 6.14. When the wind direction is discretized into a higher number of sectors, e.g. 24, 36, 180 or 360, the results follow a quite stable pattern. In this case, the LCOEs from the optimization are *consistently* better than the one from the real regular layout of EL (i.e. there are no "jumps", as happens from less sectors) confirming the fact that irregular turbine dispositions usually perform better than the regular ones. The real layout of EL is then *fictitiously* good at 8 sectors. As can be noticed from the *red line* in Figure 6.15a and from the *orange line* in Figure 6.15b, if the outcome from the 360°-based optimization is taken as reference, the results from a rough sampling (8 to 20 sectors) are quite far from the red/orange line, whereas the results from a finer sampling (24 onwards) present a small deviation. In other words, if the finest sampling is assumed to be *correct* (because it is the most accurate obtainable sampling) and the optimization is carried out *at least* over 24 sectors, then the final outcome does not deviate too much from the most *accurate* one ($\theta = 1°$, i.e. 360 sect.).

[7] no optimization is carried out here!

- The smooth trends due to finer sampling can be explained with the help of Figure 6.16.

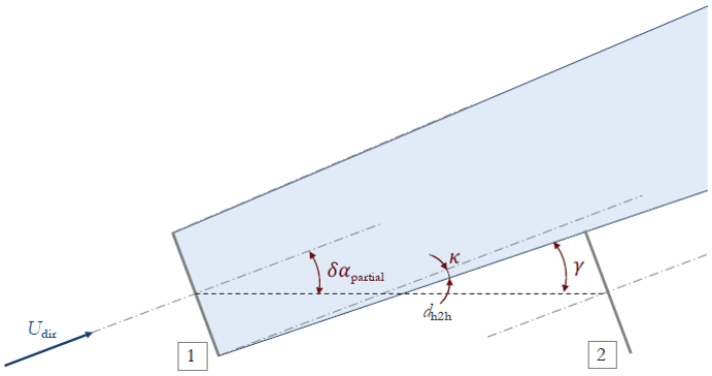


Figure 6.16: *Wake merging for two turbines [3]*

Given two turbines 1 and 2 and assuming a linear wake profile, $\delta\alpha_{partial}$ *is the angle at which the wake from turbine 1 starts hitting turbine 2.* Being the wake decay coefficient k (not shown in the figure) and D the rotor diameter, the following equations might be derived due to geometrical considerations [3]:

$$\kappa = \tan^{-1}(k) \qquad [^\circ] \tag{6.9}$$

$$\gamma = \sin^{-1}\left(\frac{D\cos(\kappa)}{d_{h_2h}}\right) \qquad [^\circ] \tag{6.10}$$

which holds:

$$\delta\alpha_{partial} = \kappa + \gamma \qquad [^\circ] \tag{6.11}$$

For angles which are lower than $\delta\alpha$, the wake incidence on turbine number 2 will be always *detected* in the discretization. This means that when a sufficient degree of precision is reached in the wind direction sampling, no further accuracy in the analysis might be achieved. If the data for the case studies in this research are used, the expression for $\delta\alpha_{partial}$ becomes:

$$\delta\alpha_{partial} = \sin^{-1}\left(\frac{D\cos(\kappa)}{4D}\right) + \tan^{-1}(k) = \sin^{-1}\left(\frac{\cos(\tan^{-1}(k))}{4}\right) + \tan^{-1}(k) \qquad [^\circ] \tag{6.12}$$

The maximum value of $\delta\alpha_{partial}$ below which no further improvement is achieved is around 17° (with k between 0.04 and 0.06). From the figures before, it can be seen that the lines are smoother starting from 24 sectors, i.e. a value of $\theta = 15°$. A rough wind direction sampling is the reason for the occurrence of the irregular patterns which occur between 8 and 20 sectors. This indicates that a layout which has been optimized for an accurate direction sampling provides good results *consistently*, e.g. the solution from a 36-sector based optimization is also good for 180 or 360 sectors. From the previous points, it is evident that a more precise wind discretization helps the user *trust* the results from the optimization. It must be noted that the value of d_{h_2h} in Equation 6.12 has been put equal to $4D$ as this is the minimum distance the turbines are allowed to have between each other. If this distance is higher, the value of $\delta\alpha_{partial}$ *decreases*. To provide an example, when $d_{h_2h} = 5D \rightarrow \delta\alpha = 14°$; if $d_{h_2h} = 6D \rightarrow \delta\alpha = 11°$ and if $d_{h_2h} = 20D \rightarrow \delta\alpha = 5°$. This means that 24 sectors is the *minimum* allowable degree of accuracy which should be considered when performing OWFLO. In fact, if the turbines were placed further from each other, as it would happen in a big OWF area, then there would be the necessity to use a finer sampling (180° to 360°) in order to detect some additional wake effects.

From the observations listed above, *it is recommended, for the future user, to sample the wind direction at least into 24 sectors as this choice provides very similar results to those coming from more sectors.* Indeed, the computational time plays a big role here. Though being higher than an 8-sector based optimization, it is still reasonable for θ above 10°, while it becomes pretty unsustainable (order of magnitude $\approx$ 24 hrs.) when more sectors are considered.

As a last remark, the same analysis as Feng and Shen is carried out [2]. This consists of *re-evaluating* the layouts coming from the sector-based optimization over *all* the other samplings. For example, the layout from the 8 sector-based optimization by means of GA is re-computed for *all* the other sectors. This is done in order to see whether the optimization relative to one stepsize may lead to a bad result if re-evaluated over another stepsize.

In their paper, Feng and Shen select the sampling which overall gives the highest improvements consistently, as shown in Figure 6.17.

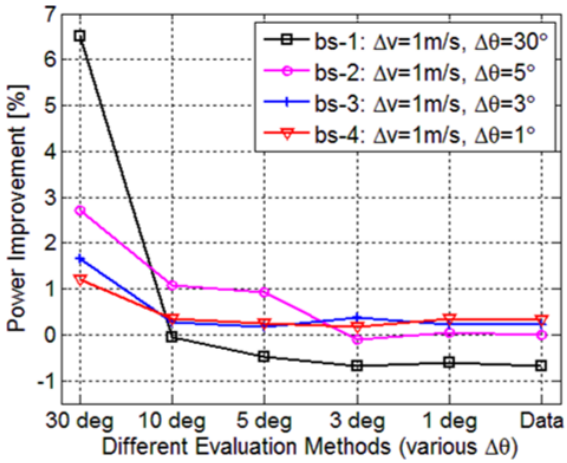
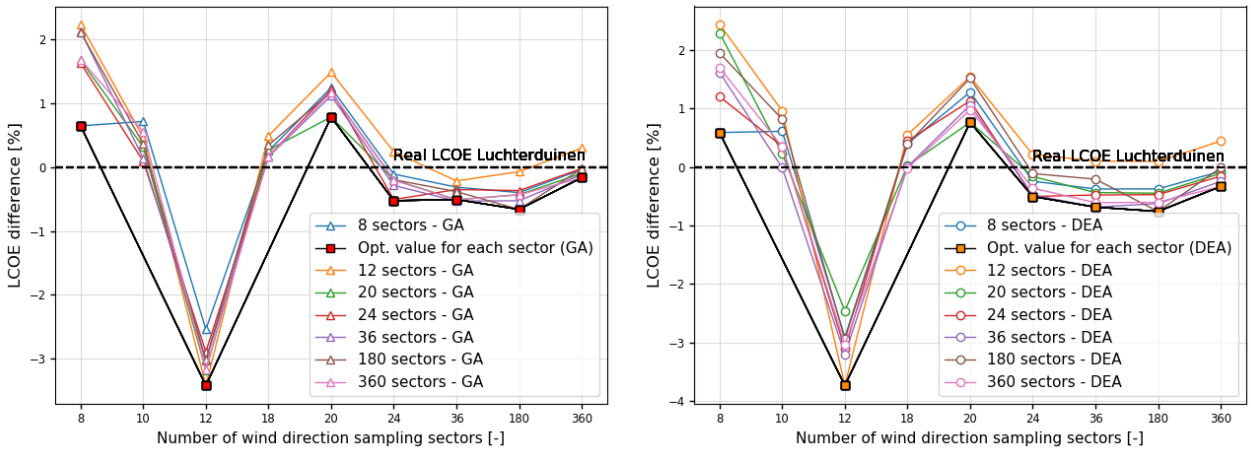


Figure 6.17: *Sensitivity analysis of the optimization. The improvement in one case may give worse results if evaluated over other cases [2]*

From the figure above, it is evident that the result which consistently gives an improvement in the power output is the one calculated with a sampling of $\theta = 1°$. (In Figure 6.17, the red line: the black line, for instance, yields a better output only if re-evaluated over its own original sampling of 30°, while it fails to achieve an improvement for a finer sampling). The authors thus recommend that choice to discretize the wind directions.

When this analysis is applied to this research, the outcome is shown in Figures 6.18a and 6.18b.



(a) *GA* **(b)** *DEA*

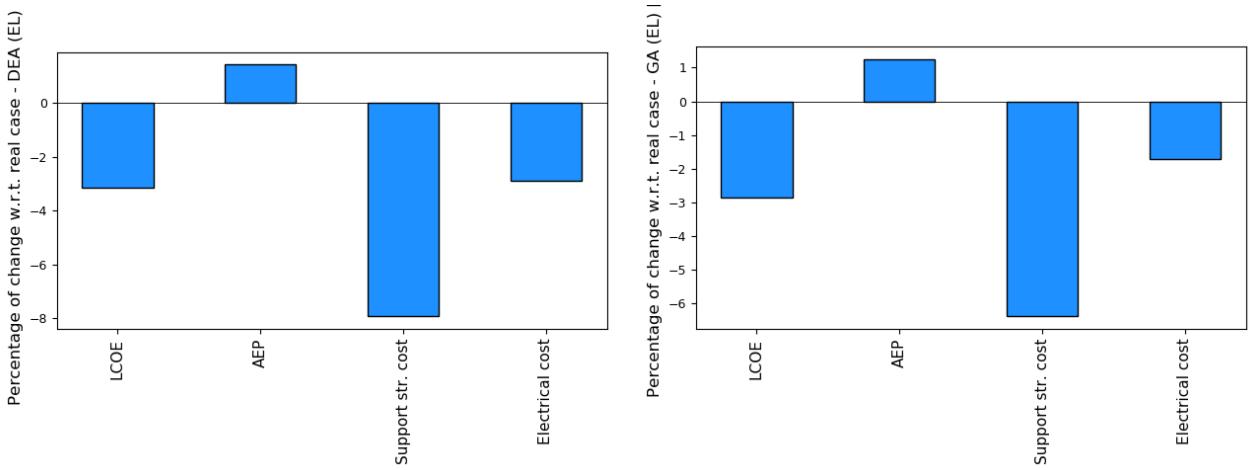
Figure 6.18: *Re-evaluation of the optimization from Figs 6.15a and 6.15b over all the other samplings*

As can be seen, the difference among the re-evaluations is not that high, meaning that an optimization which is based on a low number of sectors *may still obtain a good layout if re-evaluated over 180 or 360 sectors.* This might be useful to avoid the long simulations which occur when performing the optimization over more sectors. However, two observations should be done: firstly, the re-evaluation of the 12 sector-based optimization (orange line) yields worse results for more stepsizes; secondly, as the black line in both plots shows[8], the sector-based optimization obtains the *lowest* (i.e. best) LCOE values. In other words, the best result for e.g. the finest sampling ($\theta = 1°$) can only be obtained when optimizing over 360 sectors; and more, the best result for e.g. 20 sectors can be achieved only by optimizing w.r.t. 20 sectors. All in all, a good outcome can still be found when the algorithms work with less sectors and the resulting layout is re-computed, but the user should keep in mind that no evidence can be given in advance about the goodness of the result (as the example of the 12 sector-based optimization demonstrates).

[8]The black line is the same as in Figs 6.15a and 6.15b

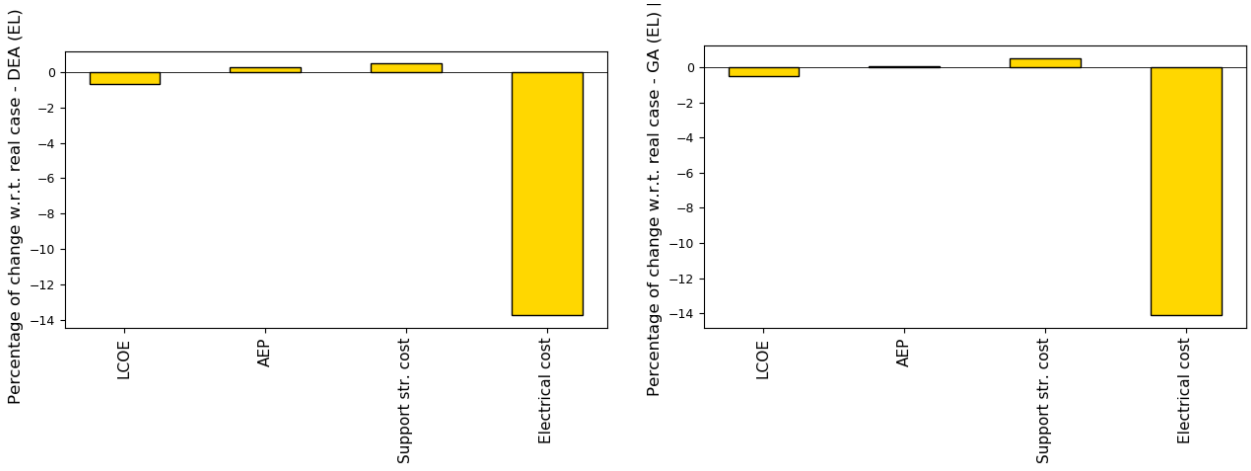
6.5 Influence of the design areas on the optimization

The objective function which was considered for the optimization is the LCOE. This is because, as stated in Paragraph 2.3.1, its definition is able to combine good and bad properties of a wind project so that they are correctly weighted [6]. Since the LCOE is made by several factors, it is convenient to analyse to what extent these factors contribute to the overall decrease of the cost of energy. The aim is trying to define what design areas mostly influence the optimization.
The bar charts below illustrate the difference in percentage between the optimized solutions and the real design for PAWP and EL[9], taking into consideration the overall levelised cost of energy and the three main design areas (aerodynamic performance, cable topology and support structures). The optimized solutions were obtained thanks to the repair mechanisms-based GA and DEA, with a wind direction sampling of 36 sectors.



(a) *DEA (PAWP)* **(b)** *GA (PAWP)*

Figure 6.19: *Mean percentage of change from real design (the line at $y = 0$) the optimized solution - PAWP*



(a) *DEA (EL)* **(b)** *GA (EL)*

Figure 6.20: *Mean percentage of change from real design (the line at $y = 0$) the optimized solution - EL*

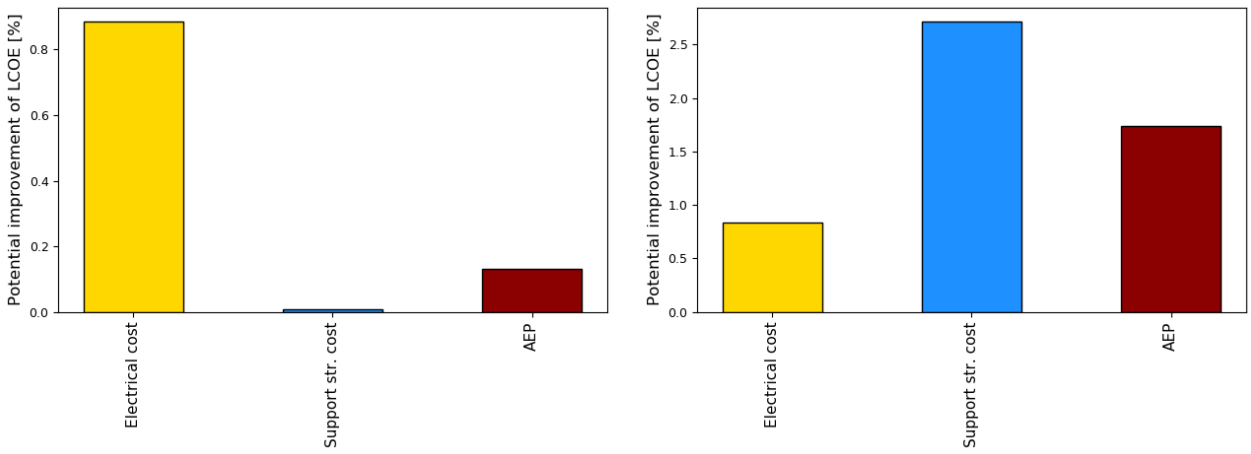
Figures 6.19a 6.19b illustrate the performance of the optimal design - from the DEA and the GA - w.r.t. the real OWF of Prinses Amalia. As can be seen, the decrease in the LCOE is significant (around 3%). The biggest improvements occur in the structural design, followed by the electrical costs and the AEP. This can be explained by taking a look at Figure 3.5, where the bathymetry in correspondence with PAWP is pretty rough, with values of the water depth ranging from 19 to 27 m; the electrical costs go down due to the clustering of the turbines in shallower regions, which interestingly does not prevent the AEP from decreasing. The reason for that might be because of the large area which is described by the boundaries of PAWP. Although the turbines move towards

[9]It is useful to remind the reader that these are the *realised* designs, i.e. the baseline designs computed in WINDOW and thus subjected to its models. They do not represent the *real* values for PAWP and EL

shallower regions, there are still margins of improvement from the aerodynamic point of view due to the vast available area.

By contrast, the biggest improvement in the LCOE for EL occurs in the cable topology. This is because the bathymetry of EL (Figure 3.3) is *flatter* than in PAWP and the wind farm area is much *tighter*. This means that no significant difference can be expected in the support structure costs when the turbines are moved throughout the design space. Therefore, the effect of the structural design on the LCOE is not supposed to be large. The optimizer is then more biased towards lowering the electrical costs. Clustering the turbines together to obtain an overall smaller cable length does not affect, for the reasons described above, the support structure costs; moreover, due to the fact that the area of EL is tight, no appreciable increase of the AEP takes place.

An important remark has to be stressed. The LCOE is not a linear function and the contributions of its parameters are not equally distributed. In addition, from the bar charts above some design areas improve more than others, *depending on the case study*. However, the previous analysis just showed *where* the biggest changes in the disciplines occurred, without quantifying their actual *influence on the LCOE*. For example, 2% in AEP increase may have a higher weight than 14% lower electrical costs. This is why some further investigation has been carried out. The factors which are used to calculate the cost of energy (AEP, C_{inv}, C_{dec}, etc.) were saved for the *real design* of PAWP and EL. After that, some new optimizations were performed for both the case studies *with different objective functions*. These were: *the AEP* (maximization of the energy yield), *the support structure cost* (minimization of the structural costs) and *the cable cost*. The results from these optimizations were then substituted into the formula of the LCOE one by one, *maintaining all the other values from the baseline design the same*. For instance, if the objective function is the AEP, only that value is substituted into the expression of the LCOE, while all the other parameters are *the same* as in the real design. The new value of the cost of energy consequently shows which are the potential margins of improvement of each discipline and to what extent the LCOE changes. This analysis is summarized by Figures 6.21a and 6.21b.



(a) *EL* **(b)** *PAWP*

Figure 6.21: *Potential contributions to the LCOE from the optimizations of distinct design areas (GA)*

From those bar charts, the following conclusions can be drawn.

- As regards Eneco Luchterduinen, Figure 6.21a is in compliance with Figures 6.20a and 6.20b. The sectors where the highest improvements occur are also the sectors which contribute to the highest extent to the overall decrease of the LCOE. This is because of the considerations about the bathymetry of EL and the narrow wind farm area which have been described above.
- As regards Prinses Amalia, when the objective function is the support structure cost, its value is the one which contributes the most to the potential LCOE improvement. Interestingly, the AEP is ranked at the second position in this case, showcasing that it has a higher potential for LCOE reduction than the electrical costs. In this case the bathymetry plays a significant role, as well as the aerodynamic performance of the wind farm, which can be enhanced considerably due to the large size of the wind farm area.

In conclusion, when the influence of the design areas on the optimization is taken into account, no general answer can be given about which sector contributes the most, as there is a difference between the two case

studies (PAWP and EL). However, *this provides further evidence of the strength of an MDAO procedure*: the multidisciplinary optimization always tries to reach a trade-off between sometimes conflicting objectives, *which are case-study dependent.*

6.6 Summary of the chapter

This chapter illustrated the results from the optimizations of two case studies: PAWP and EL. Different trends characterize the combinations of optimizing procedures that have been analysed. The best performance was achieved by the repair mechanism-based GA and DEA with a feasible initialization. The choice was made thanks to some quantitative assessment criteria and a multi-criteria analysis. After having identified the best procedure to be implemented in the driver within an MDAO workflow, the influence of the wind direction sampling stepsize has been investigated. It has been found that when the number of sampling sectors is above 24 (corresponding to $\theta = 15°$) no further precision of the wake deficits is achieved; lastly, the influence of the design areas on the optimization was analysed. The conclusion is that no general answer can be given about what discipline mostly affects the LCOE calculation, as the results are case study-dependent. However, this provides further evidence of the strength of an MDAO procedure, as the multidisciplinary optimization always tries to reach a trade-off between sometimes conflicting objectives.

Chapter 7

A new case study: Borssele III

7.1 Introduction to the chapter

This chapter presents the analysis of a new case study. Since the main application of the MDAO workflow is to carry out the preliminary design assessment, the question which arises after the discussion of the results in Chapter 6 is whether the chosen optimizing procedures are able to deal with more complex design situations, such as a larger number of constraints or a longer design vector. Although these more challenging situations could have been fictitiously reproduced either in Prinses Amalia or Eneco Luchterduinen, it has been decided to analyse a third case study, whose financial close took place recently, Borssele III. Borssele III is an interesting project as its wind farm area is characterized by a forbidden area due to the presence of cable routes [14].
It is crucial to remark that the optimizing procedures which were selected in Chapter 6 *are based on the conditions that the wind farm layout is the design vector and that only two constraints (spacing and boundary) are considered*. Therefore, there is no evidence that their performance will be still good under more challenging conditions, as well as it could be that some of the discarded combinations may actually perform well in this situation. Nevertheless, it has been decided to stick to the chosen combinations and see how they perform in Borssele III.
An overview of the new conditions is provided in Section 7.3; after that, the results are illustrated. Moreover, an additional analysis on the effect of neighbouring wind farms has been carried out, in order to test how the layout optimization is affected when more than one wind farm are taken into account.

7.2 Borssele III: presentation of the new case study

Borssele III is a project developed by Blauwwind II, which is a consortium composed by Shell, Eneco, Van Oord and Diamond Generating Europe Limited (Mitsubishi). The project consists of 37 Vestas V164-9.5MW turbines, for a total rated power of 351.5 MW. The wind farm is 71.4 km^2 in size but its actual area is 64 km^2, due to existing cables/pipelines which cross the site [14]. This means that the available area for the development of the wind farm is split into two "allowed" zones, as shown in Figure 7.1.

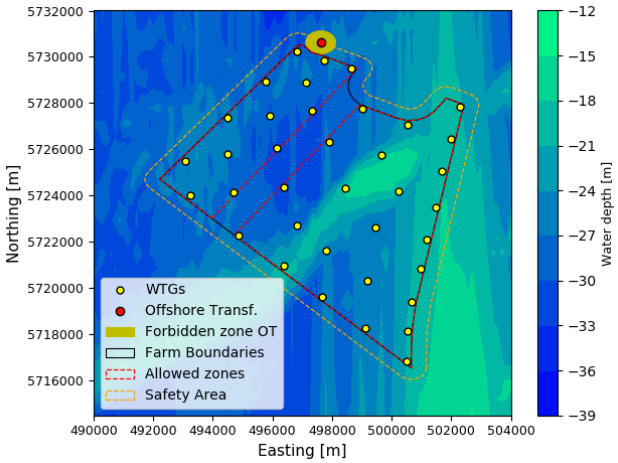


Figure 7.1: *Layout of Borssele III. The OT is 500 m far from the boundaries of the wind farm*

The offshore transformer is located outside of the wind farm, 500 m far from the boundaries. The bathymetry is quite irregular, with values of the water depth ranging from 12 to 35 m. The grid connection point is located in Vlissingen [14].

7.3 A more difficult optimization: presentation of the challenges

This paragraph illustrates the three challenges which have been chosen to test the optimizing procedures. It is crucial to note that the purpose of this analysis is not suggesting a new layout for Borssele III, but just verifying whether the optimizer is able to work under more critical restrictions.

7.3.1 Challenge 1: simple case

Challenge 1 consists of carrying out the optimization under the same conditions as PAWP and EL, i.e. layout as design vector, the LCOE as objective function and two constraints. As mentioned in Paragraph 7.2, the number of turbines to be optimally placed is fixed and equal to 37. Therefore, once that all the environmental conditions and the fixed design parameters have been inserted in WINDOW, the algorithms run with a feasible initialization and a repair mechanism. This challenge has been selected just to test the "robustness" of the optimization, i.e. the good performance of the driver with a different use case (see Paragraph 6.3 for more information).

7.3.2 Challenge 2: optimization with forbidden zones

In Challenge 2, the optimizer is called to face a more difficult situation: being the LCOE the objective function and the initialization feasible, one more constraint is added. This is a forbidden corridor due to the presence of cable routes [14]. Standards also prescribe a minimum distance from the offshore transformer (OT) of $500m$, but this is already satisfied. By using the same notation as Paragraph 2.3.3, the additional constraint can be written as:

$$g_2(\mathbf{x}) = (x_i, y_i) \notin Z \quad \forall i \in [0, 1, ..., N_T] \tag{7.1}$$

being Z the set containing all the points in the design space which belong to the restricted zone. When g_2 is considered, the repair mechanism in both the GA and the DEA should be modified accordingly. In particular, the standard deviation (σ) in Equation 5.8 has been raised to 500 m (whereas for two constraints it was set to 400 m). This is because it has been noted during the runs that if one turbine happens to be exactly in the middle of the forbidden zone, the repair mechanism takes a long time to find a new suitable place, as the probability to change the turbine position over a large distance is quite low (see Figure 5.7b).

7.3.3 Challenge 3: optimization with a variable number of turbines

Challenge number 3 considers the additional constraint from Challenge 2 and increases the difficulty of the problem by *varying the number of turbines*, i.e. the design vector is now both the $[x, y]$ coordinates of the turbines *and* their *amount*. Every individual (layout) in the population of solutions has one more dimension: $[[[x_1, y_1], [x_2, y_2], .., [x_N, y_N]], N]$, where N is the number of turbines of that individual.
Doing such an analysis with N belonging to the design vector is interesting as the choice of this value is something which is difficult to make *a priori*. A high number of turbines ensures higher costs but it is likely to increase the energy yield. Optimizing N means actually finding the optimum number of turbines such as below that value the project does not ensure enough revenues and beyond which the energy output does not increase with the same rate as the costs. However, N usually does not need to be computed, as when the concessions to build new wind farms are given by the government, the rated power of the wind farm is a targeted fixed quantity. The choice of the turbine is determined on the basis of the contracts with the manufacturer. Therefore, once again, Challenge 3 has been carried out just *to verify the strength of the optimizing procedure*.

So far, the analysis which has been done considers a *fixed* N. The user was able to *modify* this value *before starting* the optimization, but this quantity remains the same over the iterations. The way Challenge 3 has then been formulated is as follows: given a range of turbines to choose from, carry out several *separate* optimizations with a *fixed* number of turbines *which varies* in every case; then, check for what value of N the lowest LCOE occurs; in the end, perform a *all-in-one* layout optimization with a *variable* number of turbines to see whether the value of both the LCOE and N are in accordance with the results from the *separate* optimizations. If this happens, then it means that the optimizer is not only able to find a good (low) value of the LCOE, but even the optimal number of turbines automatically. Figure 7.2 provides an illustration of this strategy.

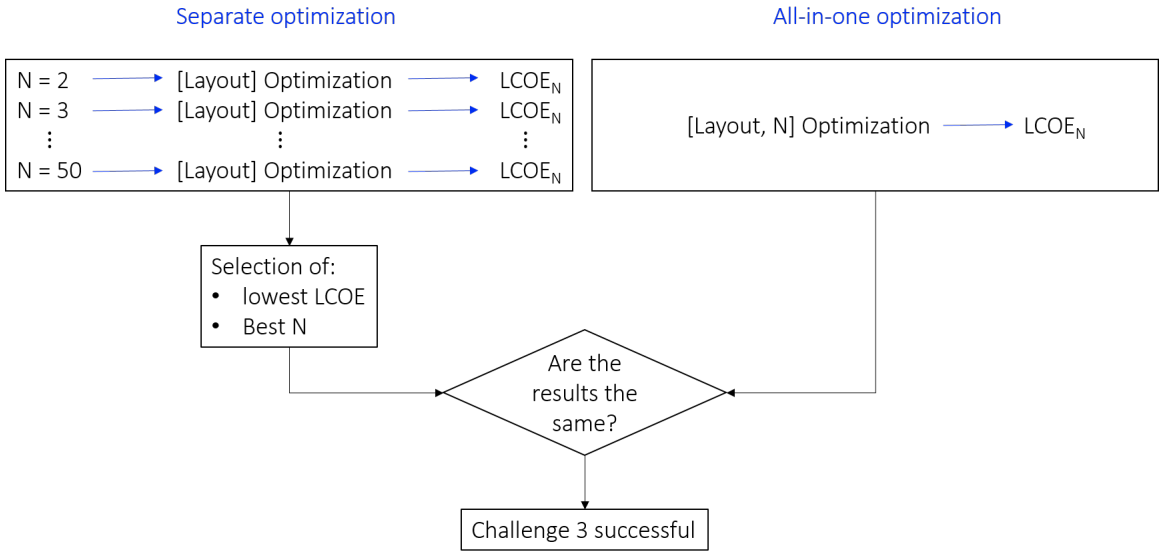


Figure 7.2: *Visualization of Challenge 3*

As can be seen, the separate optimization provides the lowest LCOE (i.e. the best layout) with the fixed N used as input; the all-in-one optimization computes everything in one run. In this research, $N \in [2, 50]$.

7.4 Results

The outcome from the first two challenges are compared to the output from the baseline design, depicted in Figure 7.1. On the other hand, the outcome of the separated vs. all-in-one optimization (Challenge 3) is displayed in Figure 7.5.

7.4.1 Challenge 1

The results from the first Challenge can be summarized by looking at Figure 7.3. Four runs have been carried out. All of them yielded a lower LCOE value w.r.t. to the realised design.

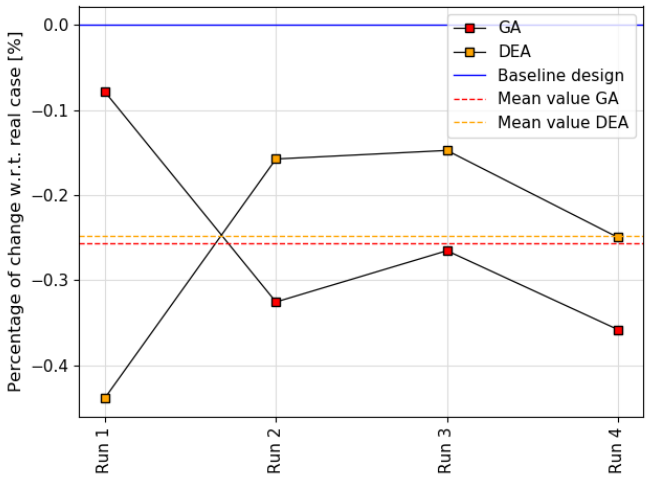


Figure 7.3: *Visualization of the results for all the runs - Challenge 1*

The figure above shows that both the DEA and GA with feasible initialization and repair mechanism dealt with Challenge 1 successfully. This is not surprising, as apart from being the combinations with the best optimality and random sampling values (Tables 6.2 and 6.3), they also have a very good robustness, i.e. a good performance for different case studies.

7.4.2 Challenge 2

The two optimizers were able to find solutions with a lower LCOE than the real design of Borssele III for the conditions of Challenge 2. Moreover, all the turbines are placed only in the allowed zones. This means that the repair mechanisms are both able to cope with the additional constraint and to yield good results.
From figure 7.4 it can be seen that the mean value of the LCOE over multiple runs for both the GA and DEA

is higher than in Challenge 1. This was expected, as in Challenge 2 the turbines have less freedom to move throughout the design space, with the consequence of a less effective exploration.

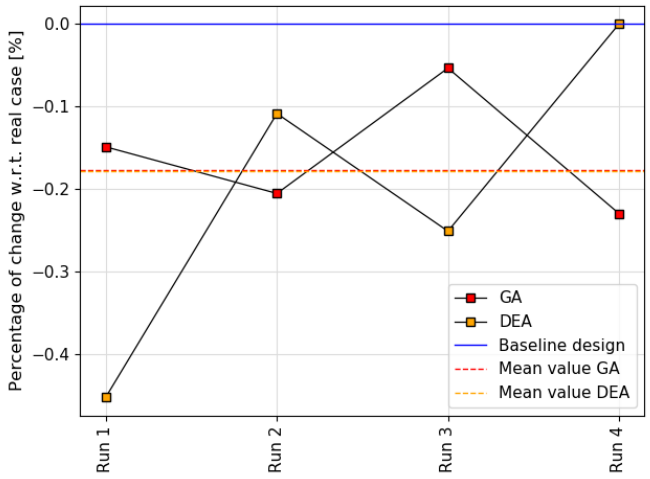


Figure 7.4: *Visualization of the results for all the runs - Challenge 2*

7.4.3 Challenge 3

The results from Challenge 3 are summarized in Figure 7.5. The blue trend represents the results from the separate optimization. As can be seen, the lowest LCOEs occur for $N = 20$, $N = 18$, $N = 21$, the minimum being in $N = 20$.
The all-in-one optimizations have been performed 6 times for each algorithm. The outcome is interesting. The all-in-one optimization *is able to yield LCOE values which are comparable to those coming from the separate strategy, while finding the same value of N.*

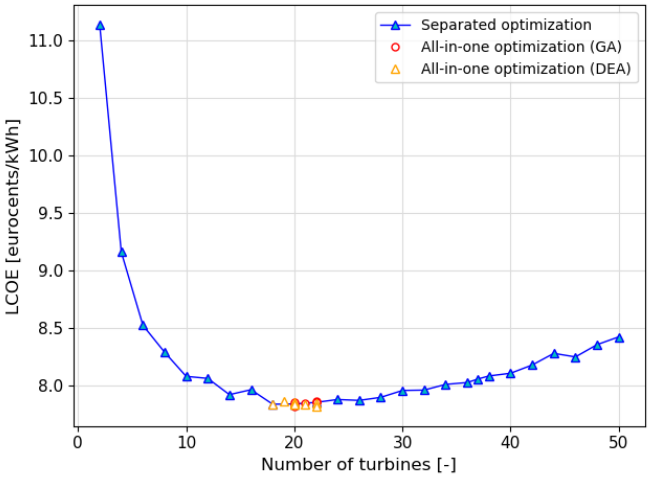


Figure 7.5: *Visualization of the results for the all-in-one optimization vs. separated optimization - Challenge 3*

The slight differences in the values of N obtained by the optimizers can be explained as follows. There is an infinite range of combinations in terms of turbines' layout. Since the implemented algorithms are *meta-heuristic*, i.e. they contain some randomness in their formulation, the objective function is *satisfactorily low*, though not being the overall global minimum. It can be therefore possible for the optimization to find combinations of "N+layout" which obtain very similar LCOE values, with different numbers and dispositions of the wind turbines. Further evidence can be obtained by looking at the values from the separate optimizations when $N \in [18, 22]$: although the overall minimum is located at $N = 20$, the magnitudes of the cost of energy are very close to each other. Therefore, in case of an all-in-one optimization, it is up to the company to decide which alternative is the most convenient, based both on qualitative and quantitative considerations. For instance, installing 18 turbines instead of 20 is advantageous for the installation and for the O&M, being the overall probability of failure lower [3]; on the other hand, a higher number of turbines yields a higher energy output, which can power more households.
It is worthwhile to point out that, as mentioned in Paragraph 7.3.3, *this analysis is not meant to suggest*

installing around 20 turbines in Borssele, rather than 37. The goal was just challenging the algorithm to find the minimum LCOE with a varying value of N. Indeed, the physical and economical models in the analysis block (WINDOW) play an important role. If some more accurate financial modeling had been included in the tool, the optimum number of turbines would have been likely to shift. From this perspective, it is recommended to adapt the cost models, as soon it is still possible, to the actual company's needs. This would enable the designer to run trustworthy all-in-one optimizations for future projects.

7.5 The effect of neighbouring wind farms

7.5.1 Overview of the problem

The last analysis which has been performed in this work is the assessment of the effect of neighbouring wind farms on the optimization procedure. Wind farms made of a large number of turbines are known to affect the local atmospherical conditions. In particular, some dense clustering creates obstacles to the entrainment of the air momentum above the turbines, limiting the recovery of the wakes. This is also called *deep array effect* [82]. Many authors investigated on the goodness of the kinematic wake models (such as the Jensen model used in this thesis) when dealing with large wind farms and/or groups of more than one OWF, to see whether these models are in accordance with real data measurements without the need to pay attention to more complex fluid-dynamic theories. Among these, Nygaard [83] carried out some very interesting research on the comparison between real data from three large wind farms (London Array, Anholt and Walney - Walney is divided into two sub-farms) and the wake deficits computed with the Jensen model, without noticing any particular "deep array effect". Conversely, Wu [84] noticed the presence of such an interaction between the turbines and the atmosphere in Horns Rev Wind Farm, and the related issues which arise with the adoption of simple kinematic models.
In this thesis, the focus is not on performing some validation of the aerodynamic models over real data, neither providing an approach to incorporate the (possible) presence of deep array effects in large wind farms. Instead, *by assuming the deep array effects to be negligible and keeping the Jensen model the same, the goal is to verify the effect of the presence of other wind farms on the layout optimization of Borssele III.* In other words, the goal is to *use* the optimizer to learn something about the consequences of neighbouring wind farms, in order to lay the foundation for further research in the future.

7.5.2 Presentation of the neighbouring wind farms

To investigate how the presence of another wind farm may affect the OWFLO, two additional plants were considered. These are the Belgian parks of Seastar and Northwind, which will be built close to the Belgian border, being in the South-West w.r.t. Borssele III. According to 4C Offshore site, Seastar counts 29 turbines, while Northwind is made of 72 windmills [85][86]. The layout of Seastar is representative for confidentiality reasons, but for the calculation the real one was used. Although the OWFs of Borssele I, II and IV could have been used, the Belgian plants were chosen because the wind has higher chances to come from the South-West.

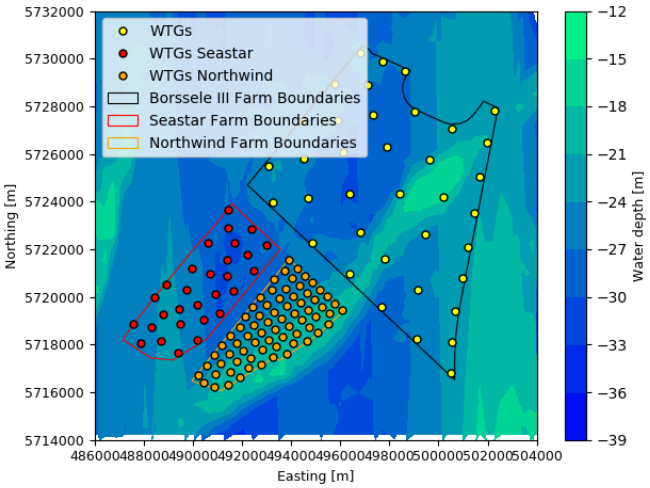


Figure 7.6: *Borssele III and the Belgian Wind Parks of Seastar and Northwind*

A remark has to be stressed. To simplify the development of the codes, the wind turbines were assumed to be the same as Borssele III, i.e. Vestas V164-9.5MW. The analysis on the effect of neighbouring wind farms is split into two parts: the former illustrates, according to the Jensen model, the difference in the power output from Borssele III both in undisturbed conditions and with the presence of Seastar and Northwind; the latter is the proper optimization. In order to better notice the influence of the aerodynamic design, *both the AEP and later on the LCOE are taken as objective functions*, using the *spacing* and the *boundary* constraints for Borssele III.

7.5.3 The energy production in disturbed and undisturbed conditions

With the help of Figure 7.7, the turbines which have been chosen to test the difference in energy production are those which are close to the South-Western boundary. These are highlighted with the purple squares.

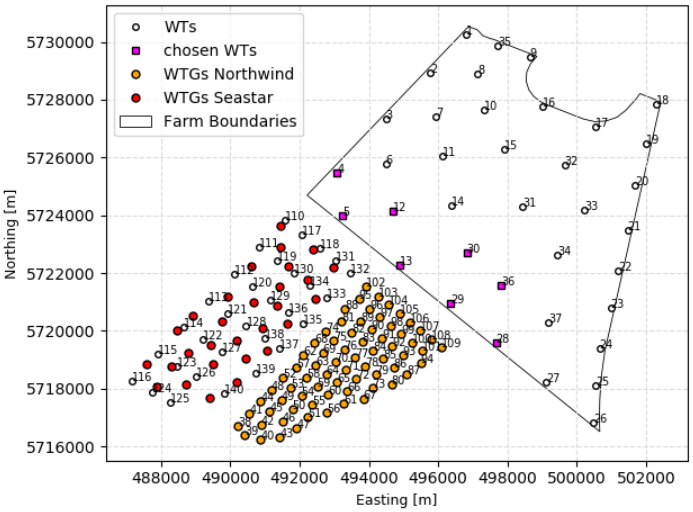
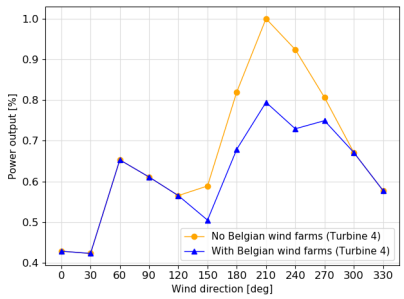
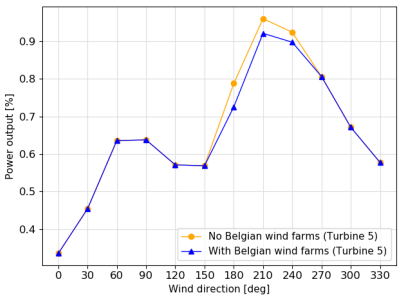


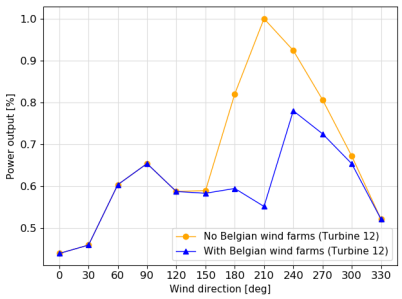
Figure 7.7: *Borssele III and the Belgian Wind Parks of Seastar and Northwind - the chosen turbines are highlighted in purple*



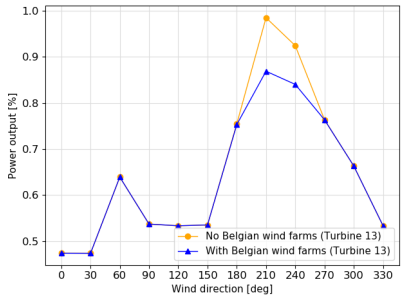
(a) *Turbine n° 4*



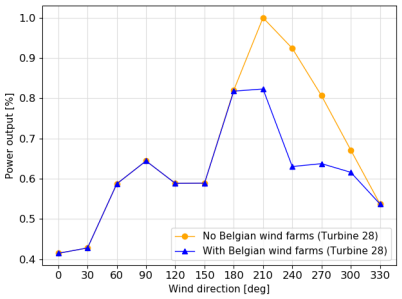
(b) *Turbine n° 5*



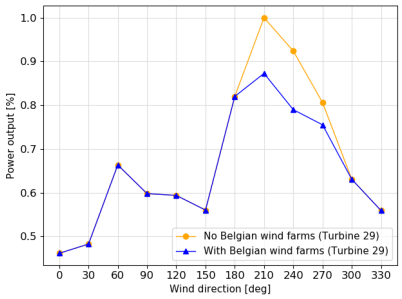
(c) *Turbine n° 12*



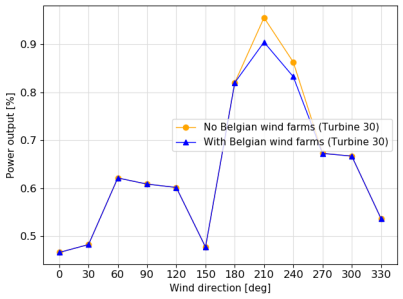
(d) *Turbine n° 13*



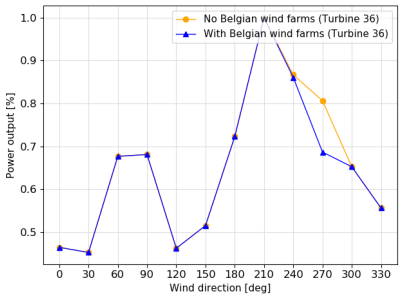
(e) *Turbine n° 28*



(f) *Turbine n° 29*



(g) *Turbine n° 30*



(h) *Turbine n° 36*

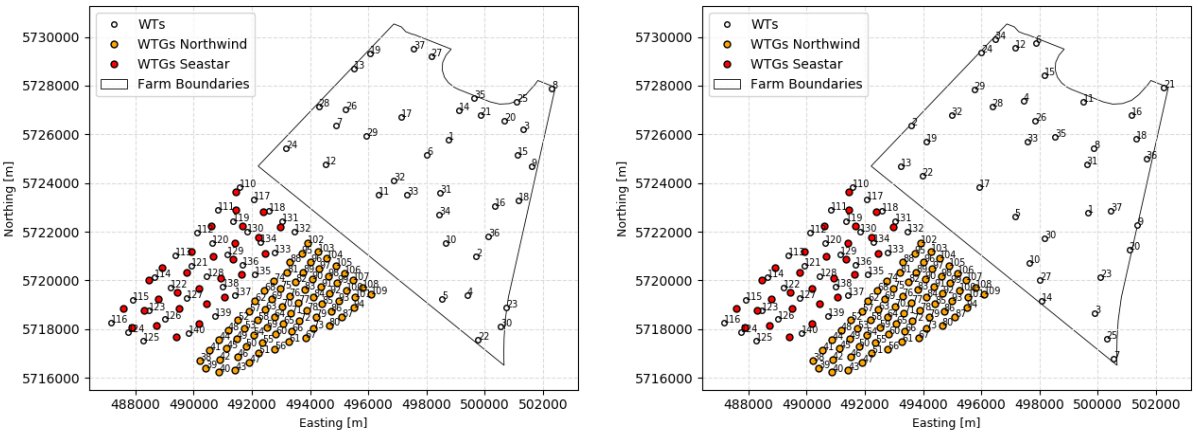
Figure 7.8: *Power output in both undisturbed and disturbed conditions for the chosen turbines*

The annual energy output w.r.t. to the wind direction from the chosen turbines has been computed in WINDOW in both the situations. The values have been normalized and plotted in the figures above. As can be clearly seen, the aerodynamic module in WINDOW *is able to detect a change in the wind (and consequently, the power output) entering the chosen turbines* when the angle of incidence is between 150 and 300°, whereas the power output for all the other sectors remains the same.
The Jensen model is known to provide good results in the *far wake region* (between 6D to 9D) [87], whereas is known not to perform well in the near wake region, which is generally acknowledged to be between 0 and 3-3.5 rotor diameters [3]. Since the distance between the wind farm of Borssele III and the Belgian plants is within this range, it can be concluded that the Jensen model implemented in WINDOW is able to yield a different energy output when another wind farm is in the surroundings.

7.5.4 The optimization with respect to AEP with neighbouring wind farms

Once that the ability of the Jensen model to detect the presence of other wind farms has been proven, the layout optimization of Borssele III w.r.t. to the annual energy production can be carried out. As was done for the three challenges before in this chapter, the optimizing procedures are the repair mechanism-based GA and DEA coupled with a feasible initialization. The resulting layouts are shown in Figures 7.9a and 7.9b. It can be clearly seen how the turbines are shifted towards the right and stretched in the direction parallel to 200°-220°, which are directions where the highest deficits used to occur (as shown in Figure 7.8).

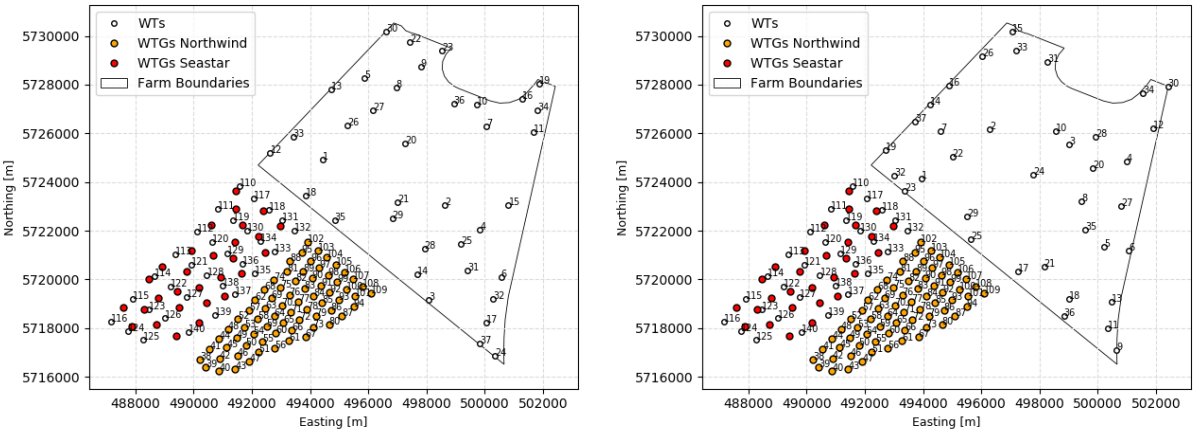
Figure 7.9: *Layout optimization w.r.t. to AEP - effect of neighbouring wind farms*



(a) *GA algorithm* **(b)** *DEA algorithm*

To further prove that the turbines' shift on the right is caused by the presence of the neighbouring wind farms, the optimization has been performed again in the *undisturbed* situation. Even if Seastar and Northwind do not appear in the optimization, the layouts are depicted together with the two Belgian wind farm to highlight the difference w.r.t. Figures 7.9a and 7.9b. The optimized layouts are shown in Figures 7.10a and 7.10b.

Figure 7.10: *Layout optimization w.r.t. to AEP - undisturbed situation*



(a) *GA algorithm (undisturbed situation)* **(b)** *DEA algorithm (undisturbed situation)*

As can be noticed, the turbines tend to cluster towards the boundaries of the wind farm, including the South-Western direction. Thus, when the influence of the two Belgian wind farms is not taken into account, the optimized layout does not feel a shift of the turbines toward the right.
The improvement in the energy yield w.r.t. to the baseline design is shown in Table 7.1.

Table 7.1: *AEP from the layout optimization of Borssele III w.r.t. to the baseline design*

	Baseline Design	**Differential Evolutionary Algorithm**	**Genetic Algorithm**
Disturbed situation	0.00	+ 0.27 %	+0.031 %
Undisturbed situation	0.00	+ 0.32 %	+0.23 %

As can be seen, the AEP improvement is higher in undisturbed conditions. This could have been expected, as in the latter situation the incoming wind is higher.
In conclusion, the Jensen model captures the difference in the power output of Borssele III when more than one farm are considered; consequently, the optimizer feels the presence of the neighbouring plants, yielding a layout which is characterized by a shift of some turbines in the opposite direction than the Belgian wind farms.

7.5.5 The optimization with respect to LCOE with neighbouring wind farms

Once that the maximization of the AEP has been done according to the Jensen model and the effect of the neighbouring wind farms has been noticed on the optimized layouts, it is interesting to try to learn something about the *consequence* of considering neighbouring plants on the levelised cost of energy. The main question to answer is whether the layout optimization w.r.t. LCOE as carried out in Challenge 1 (Paragraph 7.4) under *undisturbed conditions* still yields a better result than the baseline design if *re-evaluated* under *disturbed* conditions. In other words, whether there is the need to perform some optimization *which considers the neighbouring wind farms over the iterations* or whether the results from the *undisturbed case* can be still valuable if re-computed in disturbed conditions. In order to do that, four situations have been selected and compared. These are here itemized.

- Situation A: the LCOE of the *baseline* design in *disturbed conditions*, depicted in Figure 7.7, has been computed *only for Borssele III*, i.e. the results from WINDOW are gathered separately for Borssele III in order to obtain the LCOE of *that* farm only, tough still paying attention to the presence of the other two plants (for the AEP calculation). This case will be also named as "Disturbed baseline". It is *fundamental* to note that this LCOE *is not the same as the one used as baseline design in Figure 7.3.* In fact, when Northwind and Seastar are taken into account, less wind is available for Borssele III, as shown in Figure 7.8. The different in percentage between the baseline *undisturbed* design and Situation A is shown in the Table below.

Table 7.2: *Difference between baseline design in undisturbed and disturbed conditions*

	Baseline Design Undisturbed	**Baseline Design Disturbed**
LCOE	100%	+ 2.188 %
AEP	100%	-2.696 %

- Situation B: the best layout from the runs in Challenge 1 (Genetic Algorithm), which was computed in undisturbed conditions, is *re-evaluated* over the *disturbed* case, in analogy with Situation A for the baseline design. The LCOE of the GA-optimized layout is calculated for Borssele III by taking into consideration Seastar and Northwind. This situation will be also called "GA undisturbed - re-evaluated"
- Situation C: this is the same as Situation B but for the Differential Evolutionary Algorithm and will be referred to as "DEA undisturbed - re-evaluated".
- Situation D: a new optimization is carried out. This utilizes a GA combined with feasible initialization and repair mechanism and considers the presence of the neighbouring wind farms *in all the iterations.* This situation will be also called "Disturbed Optimization ". Only the Genetic Algorithm has been used as it is faster than the DEA: considering *all* the turbines of all the wind farms is particularly computationally expensive for the PC.

The comparison of these four situations takes place in two steps. In the former, Situation A is compared with Situations B, C and D, to see whether the results from the *undisturbed case-optimization* (i.e. Challenge 1) are *still valuable* if re-computed in disturbed conditions and whether the "Disturbed Optimization" is actually useful. The latter appraises the performance of Situation D w.r.t. Situations B and C, to show *where* the

biggest differences between undisturbed optimization and disturbed optimization lie.
The layout coming from the optimization in Situation D is shown below. It can already be seen how this is similar to Figures 7.9a and 7.9b, showcasing the important role of the AEP.

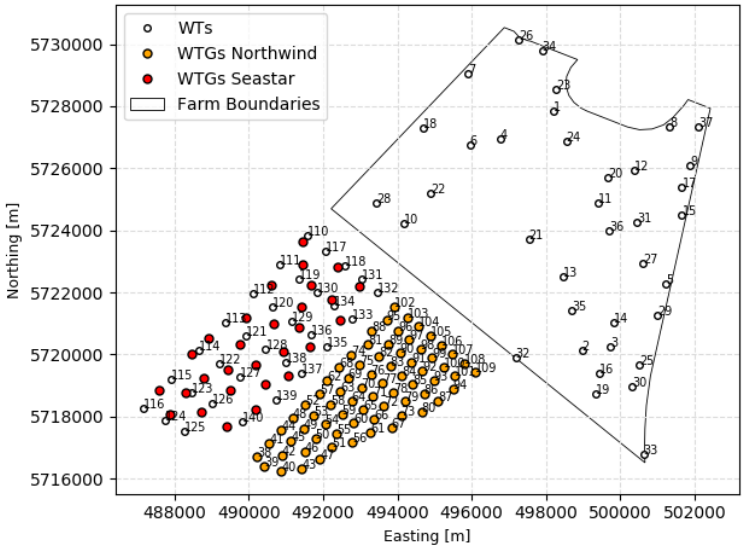


Figure 7.11: *Situation D - resulting layout. The better exploitation of the wind resource is visible*

Step 1

If the LCOE of the baseline design of Borssele III in the *disturbed case* (Situation A) is used as reference, the bar chart in Figure 7.12 illustrates the values of the LCOEs of Situations B, C and D.

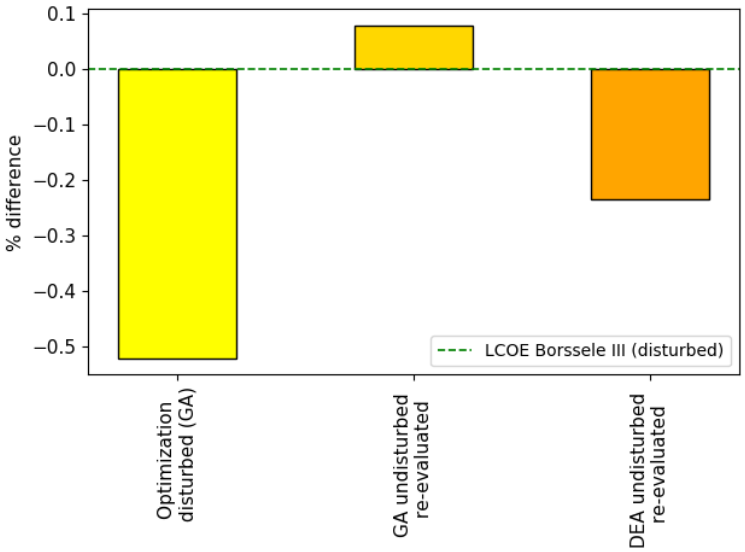


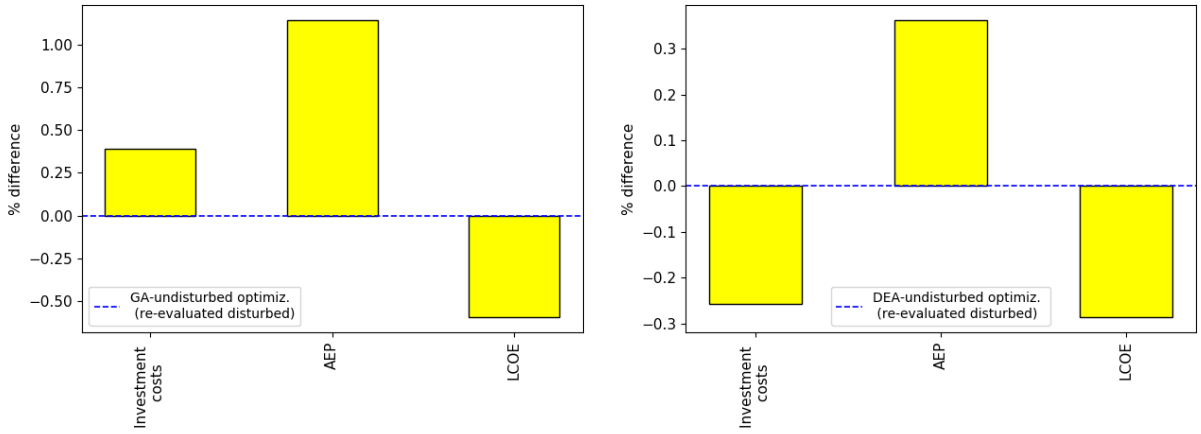
Figure 7.12: *Comparison between the Situation A (dashed line) and Situations B (in gold), C (in orange) and D (in yellow)*

As can be seen, the layout optimized by means of GA in Challenge 1 does not obtain a better (i.e. lower) LCOE than the baseline (green dotted line) if re-evaluated under disturbed conditions; by contrast, the DEA still yields a better result. On the other hand, the "Disturbed Optimization" largely outperforms Situation B and C. From this graph, it can be concluded that the results from the undisturbed optimization *might* be still good, but there is no evidence for this to happen. It seems like the DEA outperforms the GA just thanks to pure luck. The "Disturbed Optimization" obtains a better LCOE than the baseline design because they are both evaluated under disturbed conditions. This means that taking into considerations neighbouring wind farms *has the effect to shift the optimum.*

Step 2

As was shown in Step 1, the effect of neighbouring wind farm on the optimization w.r.t. the LCOE is not negligible. The layouts which had been optimized in undisturbed conditions *might* still perform better than the baseline, but they are outperformed by the layout computed in disturbed conditions. Figures 7.13a and 7.13b illustrate *where the biggest differences* between Situation B and C and Situation D come from.

Figure 7.13: *Step 2 visualization: comparison of the "Disturbed Optimization" (Situation D) with the undisturbed optimization re-evaluated in disturbed conditions (Situations B and C)*



(a) *Situation D (bars) vs. Situation B (GA - dashed line)* **(b)** *Situation D (bars) vs. Situation C (DEA - dashed line)*

As regards the GA (Figure 7.13a), the LCOE from Situation D is lower than the one in Situation B. This was evident also from Figure 7.12. This is due to the higher AEP which the Disturbed Optimization is able to yield w.r.t. the "GA undisturbed - re-evaluated", although the investment costs are slightly higher.
When Situation D is compared to Situation C (DEA algorithm) (Figure 7.13b), almost the same graph as for the GA is obtained. The LCOE is lower (as was also shown in Figure 7.12). This is caused by the higher AEP which the optimizer obtains w.r.t. to the "DEA undisturbed - re-evaluated". In this case, the AEP increase is not as high as in Figure 7.13a but this is compensated by smaller investment costs.

From this analysis, three conclusions can be drawn:

1. neighbouring wind farms *have the effect* to shift the optimum;
2. when the optimization considers the effect of neighbouring wind farms *over the iterations* (Situation D), the result is better than the undisturbed optimization re-evaluated in disturbed conditions (Situation B and C);
3. The difference between the "Disturbed Optimization" and the "GA/DEA undisturbed - re-evaluated" mostly lies in the AEP. Situation D "feels" the less incoming wind due to the presence of neighbouring wind farms and just moves the turbines to get a higher AEP, while obtaining (almost) the same investment costs as in Situation B and C

7.6 Summary of the chapter

In this chapter, a new case study is taken into account, Borssele III. This plant is characterized by more challenging conditions due to the presence of some forbidden areas within the boundaries of the wind farm. Since companies are interested in using the MDAO procedure to carry out a preliminary design analysis, it is important to check whether the optimizing strategies in the *driver* are able to yield satisfactory results in more complex design situations. Therefore, three challenges have been designed. The first consists of the optimization in the same conditions as PAWP and EL (only two constraints, LCOE as objective function and layout as design vector); in the second case, the optimizer is called to face a situation when only some areas are allowed, as it happens in reality (three constraints, LCOE as objective function and layout as design vector); the third one is the same as the previous one but it adds a design variable, i.e. the number of turbines (all-in-one optimization). To check whether the driver is able to deal with this last challenge, the results from the all-in-one optimization have been compared to the results from the separate optimization, in which the (varying) number

of turbines is fixed in every run. All the challenges highlight the ability of the optimizer to cope with more complex design requirements. On a later stage, the effect of neighbouring wind farms (Seastar and Northwind - Belgian border) on the AEP of Borssele III has been investigated. By neglecting the so called "deep array effect" and maintaining the Jensen model, the driver is able to feel the difference in the power output when more than one wind farm are considered. The result is that the algorithm tries to push the turbines of Borssele away from the (fixed) windmills of the neighbouring wind farms. When layout optimization w.r.t. the LCOE is taken into account, the optimizations carried out in Challenge 1 can no longer guarantee a better layout w.r.t. the baseline if re-evaluated in disturbed conditions. Therefore, in case it is known that there are (or will be) neighbouring wind farms and this is taken into account during the design, it is suggested using an optimization under disturbed conditions.

Chapter 8

Conclusion and recommendations

8.1 Conclusions

Within the MDAO domain, two components are identified. The former is called *analysis block* and it comprehends all the modular tools which refer to a specific physical/economic discipline (the WINDOW tool); the latter is the *driver*, i.e. an optimizing procedure which calls the analysis block in each iteration [9].

This thesis focuses mostly on the analysis of the *driver* within an MDAO procedure. As mentioned throughout the report, the driver is the *optimizing strategy*, which is made of *initialization* of the optimization (i.e. the initial guess), the way constraints are implemented (*constraint-handling techniques* - CHT) and dealt with and the choice of the *algorithm*. No clear guidelines exist to give the designer any clue about *what optimizing strategy performs best when dealing with OWFLO*. Therefore, several combinations have been tried. Six different initializations[1] have been coupled with three algorithms (GA, PSO and DEA) and four CHTs, i.e. static, dynamic, adaptive penalty functions and a repair mechanism.

- The first question this research aims to answer is: *what is the best optimizing routine - in terms of initialization, CHT and choice of the algorithm - to be coupled with the analysis block within an MDAO procedure which can be satisfactorily applied to the future design procedure of engineering companies?*
  Several quantitative assessment criteria have been defined by the author. These evaluate the goodness of each chosen combination over two case studies, i.e. Eneco Prinses Amalia Wind Park and Eneco Luchterduinen. Among all the analyzed combinations, the best performances were achieved by a Differential Evolutionary Algorithm (DEA) and a Genetic Algorithm (GA) coupled with a *feasible initialization* and a *repair mechanism* as constraint-handling technique. The PSO performs quite well when coupled with a repair mechanism, though obtainining a lower score than the other two. As already explained in Paragraph 6.2.2, the DEA and the GA are *evolutionary-based* algorithms, i.e. the entire population *converges* towards the minimum as it is part of the evolutionary process; on the other hand, although some authors rightly claim that the strongest aspect of the PSO is the collaborative research carried out by the particles [56], in the PSO the individuals are more independent from each other than the GA or the DEA as they only "feel" the best local and global position found that far. This smaller interaction between the individuals in the population *may be* the reason why the PSO has a worse performance than the other algorithms. By constrast, the GA and the DEA obtained the best scores. It is worthwhile to point out that the DEA has a higher optimality than the GA. This can be explained due to the crossover-based nature of the genetic algorithms. Since the GA combines the chromosomes inside the individuals *without creating new chromosomes*, it works with a "closed group" of variables, while the DEA has more freedom to explore the design space and thus find better LCOE values.
  After having identified the best procedure to be implemented into the driver, the influence of the wind direction sampling stepsize has been investigated. It has been found that when the number of sampling sectors is above 24 (corresponding to $\theta = 15°$) no further precision of the wake deficits is achieved. In other words, when a sufficiently high angle discretization is used, the results from the optimization are more trustful because the turbines belonging to a wake are always *detected*. However, as was already noted in

[1] Random, random-modified, equilateral triangled grid, random triangled grid, square grid and feasible

6.4.2, 24 sampling sectors is the *minimum allowable* degree of accuracy which should be considered when performing OWFLO. Indeed, it works fine as soon as the turbines are more clustered together, something that happens in tight OWF areas; by contrast, the minimum angle stepsize which is needed to detect a turbine belonging to a wake *decreases* (so the number of sector increases) when the turbines are quite far from each other, as it happens for big OWFs.

- The second research question to answer was: *which design areas - aerodynamic wake models, support structure design and cable topology - mostly affect the optimization and to what extent?*
  No general answer can be given about what discipline mostly affects the LCOE calculation, as the results are case-study dependent. As regards Eneco Luchterduinen, the sector which influences the optimization the most is the cable topology due to the relatively flat bathymetry and the narrow wind farm area; as regards Prinses Amalia, the support structure design and the AEP have the most prominent effect on OWFLO. Due to the presence of this misalignment, it is impossible to say *a priori* which discipline will bias the result the most. However, this provides further evidence of the strength of a multidisciplinary procedure: the MDAO workflow always tries to reach a trade-off between sometimes conflicting goals, which are dependent on the case study under consideration.

- The third research question to answer was: *is the chosen optimizing procedure able to deal with more complex design situations?*
  Three challenges, characterized by an increasing difficulty, have been considered over the new case study of Borssele III. The first one is similar to the previous two case studies of EL and PAWP, i.e. same constraints, objective and design vector; the second one takes into account the presence of a forbidden zone *inside* the boundaries due to the presence of cable routes; the third one adds the number of turbines to the design vector, to see whether the optimizer is capable of finding the optimal number of turbines as well as its optimal layout. All the three challengese were successful and highlighted the ability of the driver to deal with more complex situation. This is promising in case the designer is called to face additional constraints, which are likely to occurr in real projects.

- An additional analysis on the effect of neighbouring wind farms on the layout optimization of Borssele has been done. According to the Jensen model and by adopting the same optimizing strategies as in the three challenges, it was observed that considering the presence of other plants affects the AEP. A comparison between the optimized layout in *undisturbed conditions* and the optimized layout in *disturbed conditions* highlighted that in case during the design process of an OWF it is known that there are (or will be) neighbouring wind farms, this should be taken into account *within* the optimizer, as the two optimal values are too different from each other.

8.2 Recommendations

While conducting this research, the study found out some points of interest which would be interesting to further investigate on. These are itemized below.

8.2.1 Recommendations for designers

- grid-based optimizations are fast, reliable and obtain sufficiently low LCOEs. Therefore, it is recommended to use them at a very preliminary stage of the design, to at least have an idea on what to expect when performing some layout optimization.

- due to its higher speed, the use of a genetic algorithm instead of the DEA is recommended to those users who are *slightly* less interested in performing a detailed design and prefer having something quicker with little loss of optimality.

- the wind direction sampling should be made of at least 24 sectors. It has been found that beyond this value the turbines belonging to whatever wake are always *detected*. This is important as it defines the minimum degree of accuracy which is required to perform a trustworthy optimization. However, it is advisable to use even more than 24 sectors, as when there is some large spacing among the turbines in an OWF, then the minimum angle which is needed to detect a turbine in a wake *decreases*, making a finer sampling necessary.

8.2.2 Recommendations for future research

- Some more in-depth validation of the analysis block is encouraged from the *physical point of view*. Since the wake models are well developed and multiple tools are available, it is recommended to act on the support structure design and on the cable topology module. As regards the former, the effect of *fatigue* should be taken into consideration, as well as its interaction with the turbulence intensity; as regards the latter, it would be desirable to try to include some additional considerations which are not purely based on the distance between the turbines, but even on other factors such as additional constraints. However, the aerodynamic module should not be neglected. In particular, some validation over real data per wind direction sampling sector is advised to have more adherence with reality in the subsequent optimization.

- It might be worth trying to investigate on the implementation of the *deep array effect* on the analysis block. This would allow the user to convincingly carry out some optimization in presence of neighbouring wind farms, without assuming the Jensen model to be correct. However, two bottlenecks should be considered. Firstly, a good, complete theory has not been fully developed yet; secondly, there is the risk, for the computational time, to be too high due to the large amount of calculations which are required over the iterations. From this point of view, it looks more profitable to include the *deep array effect* only when optimizing w.r.t. the energy production, in order to save some additional time.

- As already mentioned in Paragraph 6.4.1, no accurate study has been carried out in this thesis about the goodness of a regular layout w.r.t. an irregular one as a starting point for the optimization. It is important to note that the wind farm boundaries of Eneco Luchterduinen are even much tighter than in Prinses Amalia. It can be consequently possible, for a regular layout, to be better performing in a tight area as some better exploitation of the wind resource can be achieved. Literature usually agrees upon the fact that irregular-shaped layouts perform generally better than regular ones [7][37][70], but this is not a rule. *Therefore, further research is advised in order to better investigate on the effect of a regular population as a starting point for the optimization.*

- The O&M cost model implemented in WINDOW is extremely simplified. Although developing a good O&M *forecasting* cost model is a difficult task, it might be worth writing down some piece of code by using the data the company is in possession of. It is importanto to stress that the effect of a more accurate model on the layout optimization is not expected to be large. However, this would help the designer have a more precise overview of the operational expenditures (OPEX) to include in the overall cost model.

- As already mentioned in Chapter 3.2.2 and 7.4.3, it is highly recommended to include a more detailed cost model, based on data which are not normally available in literature. In particular, two levels of analysis should be considered: the *project level* and the *shareholders* level. As regards the former, it would be desirable to include the *timing of the CAPEX* vs. the start of energy production, i.e. do not make *all* the investment costs occur in year 0 of the project, as there is a gap between the capital expenditures and the energy selling. Furthermore, developing a more detailed cost model for raw material (procurement and installation) is advised.
  Regarding the shareholders level, a good suggestion would be starting with a more in-depth definition of weighted average cost of capital (WACC) instead of the real interest rate in LCOE formulation, and the inclusion of company-sensitive information, such as contracts with turbine manufacturers when defining the cost of a turbine.

References

[1] S. S. Perez-Moreno; M. B. Zaaijer; K. Dykes; K. O. Merz. *Multidisciplinary design analysis and optimization of a reference offshore wind plant*, IOP Conf. Series: Journal of Physics: Conf. Series 1037 (2018) 042004.

[2] W. Z. Shen J. Feng. *Modelling wind for wind farm layout optimization using joint distribution of wind speed and wind direction*, Department of Wind Energy, Technical University of Denmark, DK-2800 Lyngby, Denmark; 2015.

[3] C. M. Engelen. *The nonlinear effect of combining uncertainties on the energy yield of an offshore wind farm - a case study for array efficiency and availability*, TU Delft MSc thesis, 2015.

[4] Global Wind Energy Council. *Global Wind Report Chapter on Global Offshore*, 2016.

[5] S. M. Fragoso Rodrigues. *A multi-objective optimization framework for the design of offshore wind farms*, 2016, Delft.

[6] M. B. Zaaijer. *Great expectations for offshore wind turbines: emulation of wind farm design to anticipate their value for customers*, 2013, Delft.

[7] B. Divacco G. Mosetti, C. Poloni. *Optimization of wind turbine positioning in large wind farms by means of a genetic algorithm*, 1992.

[8] A. Maselis. *Layout optimization of offshore wind farms affected by wake effects, cable topology and support structure variation*, 2016, Delft.

[9] M. Zaaijer S. S. Perez-Moreno. *How to select MDAO workflows*, January 2018.

[10] D. J. Wilde P. Y. Papalambros. *Principles of optimal design - modeling and computation, 2nd edition*, 2000.

[11] M. B. Zaaijer. *Economic aspects (Introduction to Wind Energy course slides)*, 2016-2017; TU Delft.

[12] J. Wilkes et al. N. Fichaux. *Oceans of opportunity - Harnessing Europe's largest domestic energy resource*, European Wind Energy Association, Brussels; 2009.

[13] A. Ning A. P. J. Stanley, J. Thomas. *Gradient-based optimization of wind farms with different turbine heights*, National Renewable Energy Laboratory, 2017.

[14] Netherlands Enterprise Energy. *Borssele Wind Farm Zone - Wind Farm Sites III and IV*, 2016.

[15] Mathematics. `https://math.stackexchange.com/questions/2374635/point-inside-a-convex-polygon`.

[16] I. Katic N. O. Jensen. *A simple model for wind cluster efficiency*, European Wind Energy Association, Conference and Exhibition, 7-9 October; Rome, 1986.

[17] A. Crespo P. Enevoldsen R. Gomez-Elvira J. Hernandez J. Hojstrup F. Manuel K. Thomsen S. T. Frandsen, L. Chacon. *Measurements on and modelling of offshore wind farms*, Tech rep. 1996.

[18] Danish Energy Agency. *Recommendation for the fulfillment of the requirements found in technical criteria*, Tech. rep, 1992.

[19] H. A. Madsen G. C. Larsen, J. Hoistrup. *Wind fields in wakes*, 1996.

[20] J. F. Ainslie D. Quarton. *Turbulence in wind turbine wakes*, Wind Engineering, 1990, pp. 15-23.

[21] K. C. Williams L. R. Esau. *On teleprocessing system design: part II - a method for approximating the optimal network*, IBM Systems Journal, vol. 5, no. 3, pp. 142-147; 1966.

[22] J. Lysgaard J. Bauer. *The offshore wind farm array cable layout problem: a planar open vehicle routing problem*, Journal of the Operational Research Society, 2014.

[23] G. Katsouris. *Infield cable topology optimization of offshore wind farms*, TU Delft MSc thesis, September 2015.

[24] J. Kollowitz. *Defining the wake decay constantc as a function of turbulence intensity to model wake losses in onshore wind farms*, Uppsala, 2016.

[25] DNV. *Guidelines for the design of wind turbines*, 2002.

[26] NEN-EN-IEC 61400-3. *Wind turbines - Part3: design requirements for offshore wind turbines*, 2009.

[27] R. P. Brent. *Algorithms for minimization without derivatives*, 1973; Englewood Cliffs, NJ: Prentice-Hall.

[28] J. Douglas Faires R. L. Burden. *Numerical Analysis*, Ninth edition, 2005.

[29] `https://www.investopedia.com/terms/w/wacc.asp`. *Weighted average cost of capital.*

[30] *Eneco Luchterduinen Offshore Wind Farm in Noordwijk*, `https://www.power-technology.com/projects/eneco-luchterduinen-offshore-wind-farm-noordwijk/`2017.

[31] Vestas Brochure. *General Specifications V112-3.0 MW 50/60 Hz.*

[32] *Eneco Princess Amalia Offshore Wind Farm Project*, `https://www.power-technology.com/projects/princess-amalia/`, 2017.

[33] Vestas Brochure. *General Specifications V80-2.0 MW 50/60 Hz.*

[34] A. Vire'. *Aerodynamic theory of wind turbines*, "Introduction to Wind Energy" course slides; Delft, 2016.

[35] A. Daneshbodi. *The effect of wake models and environmental conditions on wind farm layout optimization*, MSc thesis, TU Delft (2018).

[36] X-S. Yang. *Metaheuristic Optimization*, 2011.

[37] M. M. Abdullah S. A. Grady, M. Y. Hussaini. *Placement of wind turbines using genetic algorithms*, Renewable Energy 30(2):259-270; 2005.

[38] P. Noghreh A. Emami. *New approach on optimization in placement of wind turbines within wind farm by genetic algorithms*, 2009.

[39] G. Yang X. Li X. Zhang C. Wan, J. Wang. *Optimal micro-siting of wind turbines by genetic algorithms based on improved wind and turbine models*, 2009.

[40] G. Yang X. Zhang C. Wan, J. Wang. *Optimal micro-siting of wind farms by Particle Swarm Optimization*, Tsinghua University, Beijing; 2010.

[41] A. Bengin B. Rasuo. *Optimization of wind farm layout*, University of Belgrade, January 2010.

[42] S. Okdem D. Karaboga. *A simple global optimization algorithm for engineering problems: differential evolution algorithm*, 2004.

[43] J. Lampinen. *A constraint-handling approach for the differential evoluation algorithm*, Lappeenranta University of Technology, Finland; 2002.

[44] G. C. Larsen A. Tesauro, P-E. Rethore'. *State of art of wind farm optimization.*

[45] U. Bodenhofer. *Genetic Algorithms: theory and application*, third edition, 2004.

[46] M. C. Ferris E. J. Anderson. *Genetic Algorithms for combinatorial optimization: the assembly line balancing problem*, 1991 (revised on Januray 1993).

[47] V. Mallawaarachchi. *Introduction to Genetic Algorithms - Including Example code*, 2017.

[48] `https://nl.mathworks.com/help/gads/how-the-genetic-algorithm-works.htmlf6199`. *How the genetic algorithm works*, 2018.

[49] R. Eberhart J. Kennedy. *Particle Swarm Optimization*, 1995.

[50] `http://www.swarmintelligence.org/tutorials.php` PSO tutorial.

[51] R. Eberhart Y. Shy. *Parameter selection in Particle Swarm Optimization*, 1998.

[52] W. G. Macready D. H. Wolpert. *No free lunch theorems for optimization*, IEEE Transaction on evolutionary computation, I, 67-82 (1997).

[53] D. Filko E. K. Nyarko, R. Cupec. *A comparison of several heuristic algorithms for solving high dimensional optimization problems.*, Osijek, Croatia; 2014.

[54] S. Salsa M. Bramanti, C. Pagani. *Analisi Matematica 2*, 2013, Zanichelli.

[55] L. Johanning M. Khorasanchi S. Barbouchi A. Pillai, J. Chick. *Comparison of osshore wind farm layout optimization using a genetic algorithm and a particle swarm optimizar*, ASME 2016 35th International

Conference on Ocean, Offshore and Arctic Engineering; OMAE2016; June 19-24, 2016, Busan, South Korea.

[56] J. Kennedy M. Clerc. *The particle swarm - explosion, stability and convergence in a multidimensional complex space*, IEEE Transactions on evolutionary computation, vol. 6, No. 1; February 2002.

[57] Y. Shi R. C. Eberhart. *Comparing inertia weights and constriction factors in particle swarm optimization.*

[58] A. K. Qin B. Kazimipour, X. Li. *Effects of population initialization on differential evolution for large scale optimization*, Congress on Evolutionary Computation, July 6-11; Beijing, China; 2014.

[59] A. K. Qin B. Kazimipour, X. Li. *Initialization methods for large scale global optimization*, Congress on Evolutionary Computation, June 20-23; Cancun, Mexico; 2013.

[60] O. Kramer D. Luckehe, M. Wagner. *Constrained evolutionary wind turbine placement with penalty functions.*

[61] R. Guanche B. Perez, R. Mingues. *Offshore wind farm layout optimization using mathematical programming techniques*, Renewable Energy 53 (2013), pp 389-399; 2013.

[62] X. Zhang J. Wang, X. Li. *Genetic Optimal Micrositing of Wind Farms by Equilateral-Triangle mesh*, Tongji University; Tsinghua University, China.

[63] Z. Michalewicz S. Koziel. *Evolutionary algorithms, homomorphous mappings and constrained parameter optimization*, 1999, Massachussets Institute of Technology.

[64] C. A. Coello. *Constraint-handling techniques used with evolutionary algorithms*, Av. IPN No. 2508, Col. San Pedro Zacatenco; Mexico, D.F. 07360 (2007).

[65] X. Yao T.P. Runarsson. *Constrained evolutionary optimization - the penalty function approach*, 2002.

[66] D. W. Coit A. E. Smith. *Penalty functions*, Department of Industrial Engineering, University of Pittsburgh, Pennsylvania; 1996.

[67] R. Eberhart X, Hu. *Solving constrained nonlinear optimization problems with particle swarm optimization*, Purdue University; Department of Biomedical Engineering, Department of Electrical and Computer Engineering.

[68] W. Z. Shen J. Feng. *Solving the wind farm layout optimization problem using random search algorithm*, Technical University of Denmark, 2015.

[69] A. Paniagua-Tineo L. Prieto-A. Portilla-Figueras B. Saavedra-Moreno, S. Salcedo-Sanz. *Seeding evolutionary algorithms with heuristics for optimal wind turbines positioning in wind farms*, 2010.

[70] Z. Song A. Kusiak. *Design of wind farm layout for maximum wind energy capture*, 2009.

[71] M. J. Wichura. *The percentage points of the normal distribution*, Applied Statistics, 37, 477-484.

[72] C. Chen M. Soltani-Z. Chen P. Hou, W. Hu. *Optimization of offshore wind farm layout in restricted zones.*

[73] U. Akyazi L. Aksoy. *Advanced topics in computer sciences genetic programming.*

[74] A. Abraham M. Pant, R. Thangaraj. *Low discrepancy initialized particle swarm optimization for solving constrained optimization problems*, Fundamenta Informaticae 95 (2009) 121, DOI 10.3233/FI-2009-186, IOS Press.

[75] P. N. Suganthan J. J. Liang. *Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism*, In IEEE Congress on Evolutionary Computation (2006).

[76] A. W. Mohemmed K. A. Aziz N. A. A. Aziz, M. Y. Alias. *Particle swarm optimization for constrained and multiobjective problems: a brief review*, 2011 International Conference on Management and Artificial Intelligence IPEDR vol.6 (2011) IACSIT Press, Bali, Indonesia.

[77] A. Celen. *Comparative analysis of normalization procedures in TOPSIS method with an application to Turkish Deposit Banking Market*, INFORMATICA, 2014, Vol. 25, No. 2, 185208.

[78] C.F. Zhang J.H. Zhao J.-J. Wang, Y.-Y. Jing. *Review on multi-criteria decision analysis aid in sustainable energy decision-making*, School of Energy and Power Engineering, North China Electric Power University, China; 2009.

[79] L. M. Camarinha-Matos N. Vafaei, R. A. Ribeiro. *Normalization techniques for multi-criteria decision making: analytical hierarchy process case study*, IFIP International Federation for Information Processing, 2016.

[80] R. Ginevicius. *Normalization of quantities of various dimensions*, Journal of Business, Economics and Management, 9:1, 79-86; 2008.

[81] C.-H. Chen F. Porté-Agel, Y.-T. Wu. *A numerical study on the effect of wind direction on turbine wakes and power losses in a large wind farm*, Wind Engineering and Renewable Energy Laboratory (WIRE), Ecole

Polytechnique Fédérale de Lausanne (EPFL), EPFL-ENAC-IIE-WIRE, Lausanne CH-1015, Switzerland; 2013.

[82] *R. J.A.M. Stevens, D. F. Gayme, C. Meneveau.* Generalized coupled wake boundary layer model: applications and comparisons with field and les data for two wind-farms, Wind Energ., 19: 2023-2040.

[83] N. G. Nygaard. *Wakes in very large wind farms and the effect of neighbouring wind farms*, 2014 J. Phys.: Conf. Ser. 524 012162.

[84] F. Porte-Angel Y.-T. Wu. *Modeling turbine wakes and power losses within a wind farm using LES: An application to the Horns Rev offshore wind farm*, Renewable Energy 75, 945 (2015).

[85] Northwind Wind Farm. `https://www.4coffshore.com/windfarms/seastar-be-be06.html`.

[86] Seastar Wind Farm. `https://www.4coffshore.com/windfarms/seastar-be-be06.html`.

[87] A. Raheem-Y.-K. Wu R. Shakoor, M. Yusri Hassan. *Wake effect modeling: a review of wind farm layout optimization using Jensen's model*, Renewable and Sustainable Energy Reviews 58 (2016) 10481059.

[88] Eneco B. V. Wind. *Eneco Databreeze (private website).*

[89] M. B. Zaaijer. *Wind climate and energy production ("Introduction to Wind Energy" course slides)*, TU Delft, 2017.

[90] P. Bhattacharya. *Weibull distribution for estimating the parameters*, Kolaghat, India.

[91] K. Yoon C. L. Hwang. *Multiple attribute decision making: methods and applications*, 1981, New York: Springer-Verlag.

Appendix A

Weibull distribution

The wind data have been taken from Eneco's private database [88]. Eneco's offshore wind farms Prinses Amalia and Luchterduinen have been taken into account. In these OWFs, both the magnitude and the direction of the wind are recorded on a 10-min basis for every turbine at the hub height.
The Weibull distribution function is as follows [89]:

$$f(U) = \frac{k}{a}(\frac{U}{a})^{k-1}e^{-(\frac{U}{a})^k} \tag{A.1}$$

in which U is the independent variable representing the wind speed, a and k are respectively the *scale* and the *shape* factor. These two factors are the only things which are needed to represent the wind conditions [89].
Estimating the values of a and k starting from the measurements may not be an easy task. Several approaches exist in literature. The *Maximum Likelihood Estimator* (MLE) procedure has been chosen [90]. This is as follows.
Let $x_1, x_2, ..., x_n$ be a sample of size n (i.e. the wind speed bins) which is drawn from a probability density function (*pdf*) $f(x, \theta)$ where θ is not known. In this case, θ is a vector equal to: $\theta = [a, k]$. The *Likelihood function* is thus defined as:

$$L = \sum_{i=1}^{n} f(x, \theta) \tag{A.2}$$

The MLE of θ is the value of θ which gives the maximum L or, equivalently, the logarithm of L. The MLE is a solution of [90]:

$$\frac{d(log(L))}{d\theta} = 0 \tag{A.3}$$

If the Weibull probability density function is considered, the following expression can be written:

$$L(U_1, U_2, ..., U_n, a, k) = \sum_{i=1}^{n}(\frac{k}{a}(\frac{U}{a})^{k-1}e^{-(\frac{U}{a})^k}) \tag{A.4}$$

Taking the logarithms of equation A.4, differentiating w.r.t. a and k and equating everything to 0 yields:

$$\frac{\partial log(L)}{\partial k} = \frac{n}{k} + \sum_{i=1}^{n} log(U_i) - \frac{1}{a}\sum_{i=1}^{n} U_i^k log(U_i) = 0 \tag{A.5}$$

and

$$\frac{\partial log(L)}{\partial a} = -\frac{n}{a} + \frac{1}{a^2}\sum_{i=1}^{n} U_i^k = 0 \tag{A.6}$$

On eliminating a between these two equations and simplifying, the output is as follows:

$$\frac{\sum_{i=1}^{n} U_i^k log(U_i)}{\sum_{i=1}^{n} U_i^k} - \frac{1}{k} - \frac{1}{n}\sum_{i=1}^{n} log(U_i) = 0 \tag{A.7}$$

This complicated way of writing Equation A.7 can be actually represented in the form:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{A.8}$$

which can be solved iteratively. In fact:

$$f(k) = \frac{\sum_{i=1}^{n} U_i^k log(U_i)}{\sum_{i=1}^{n} U_i^k} - \frac{1}{k} - \frac{1}{n}\sum_{i=1}^{n} log(U_i) \tag{A.9}$$

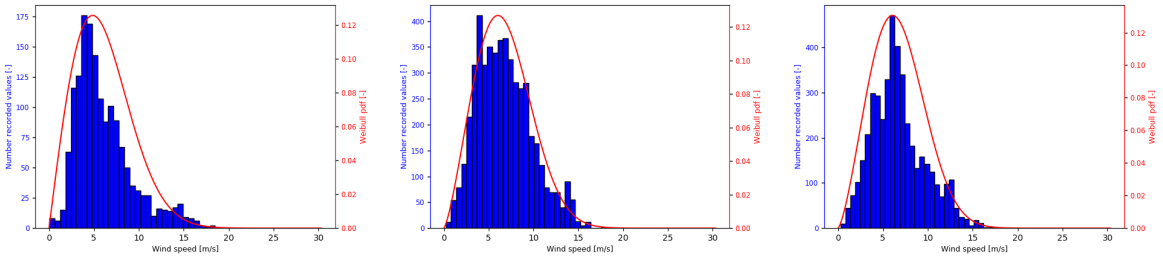
and

$$f'(k) = \sum_{i=1}^{n} U_i^k (log(U_i)^2) - \frac{1}{k^2}\sum_{i=1}^{n} U_i^k (klog(U_i) - 1) - (\frac{1}{n}\sum_{i=1}^{n} log(U_i)\sum_{i=1}^{n} U_i^k log(U_i)) \tag{A.10}$$

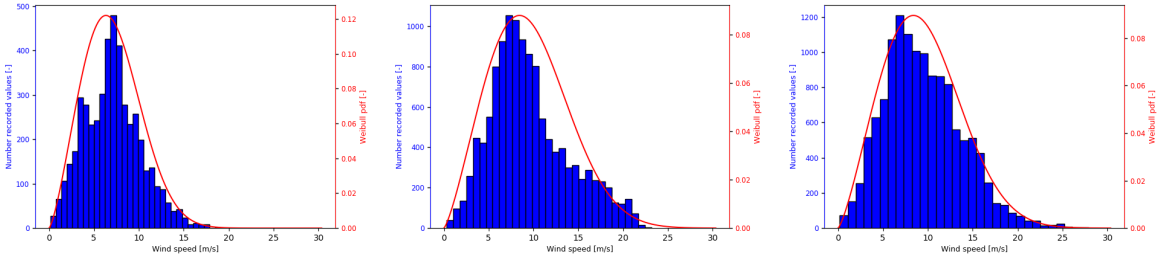
Once that k is found, a is equal to:

$$a = \frac{\sum_{i=1}^{n} U_i^k}{n} \tag{A.11}$$

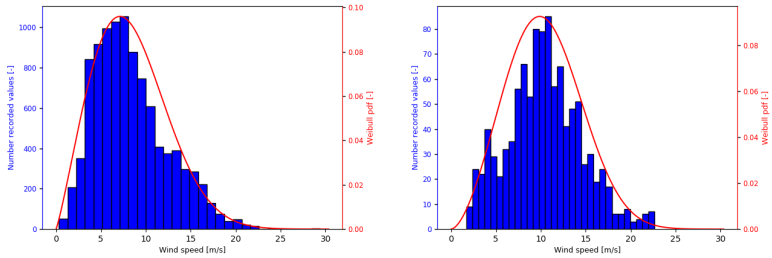
This procedure maximizes the Likelihood function. However, the author has decided to perform some optimization by minimizing the negative Likelihood function. This has been done through the `scipy.optimize` package available in Python. The objective function is $-log(L)$ and the shape and scale factor are found in an iterative way by the optimizer.



(a) *PAWP Weibull Distrib. Sector 0* **(b)** *PAWP Weibull Distrib. Sector 1* **(c)** *PAWP Weibull Distrib. Sector 2*

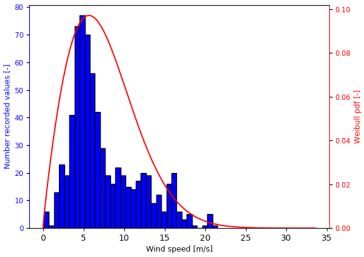


(d) *PAWP Weibull Distrib. Sector 3* **(e)** *PAWP Weibull Distrib. Sector 4* **(f)** *PAWP Weibull Distrib. Sector 5*

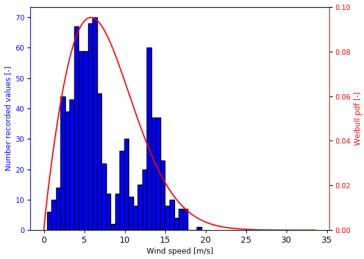


(g) *PAWP Weibull Distrib. Sector 6* **(h)** *PAWP Weibull Distrib. Sector 7*

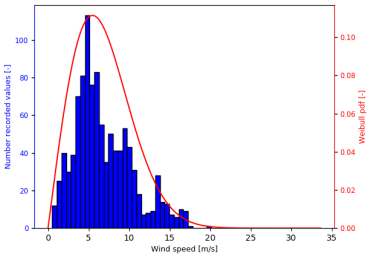
Figure A.1: *Weibull fit for Prinses Amalia: 8 windrose sectors*



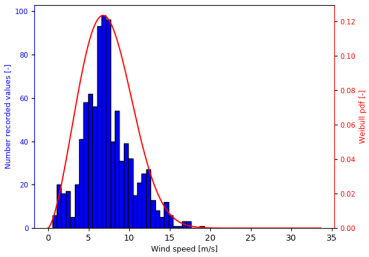
(a) *EL Weibull Distrib. Sector 0*



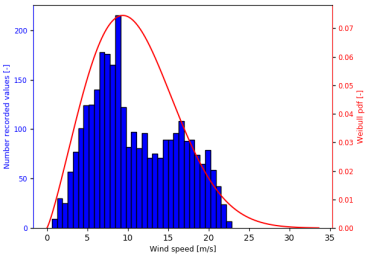
(b) *EL Weibull Distrib. Sector 1*



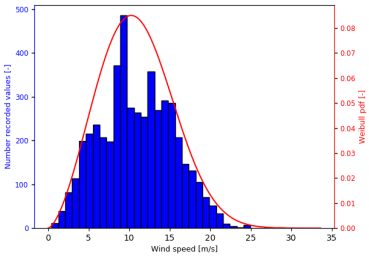
(c) *EL Weibull Distrib. Sector 2*



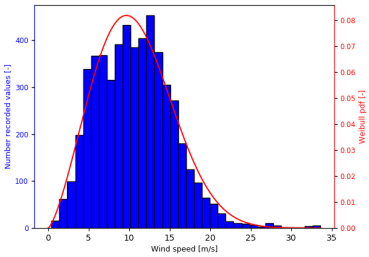
(d) *EL Weibull Distrib. Sector 3*



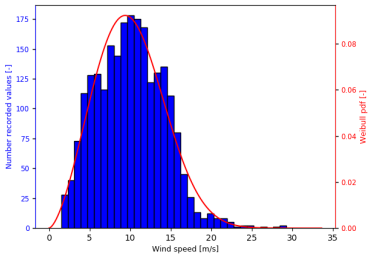
(e) *EL Weibull Distrib. Sector 4*



(f) *EL Weibull Distrib. Sector 5*



(g) *EL Weibull Distrib. Sector 6*



(h) *EL Weibull Distrib. Sector 7*

Figure A.2: *Weibull fit for Eneco Luchterduinen: 8 windrose sectors*

Appendix B

Multi-criteria analysis approach: TOPSIS

The TOPSIS procedure was originally proposed in 1981 by Hwang and Yoon [91]. In this method, each attribute - i.e. an assessment criterion - is assumed to have a tendency towards monotolically increasing utility. In words, at first Positive Ideal Solutions (PIS) and Negative Ideal Solutions (NIS) are computed. The former maximizes the "benefit" attributes and minimizes the "costs", whereas the NIS does the other way round [77]. In short, PIS gathers all the best obtainable scores while NIS takes the negative ones.

The strategy works as follows. In a MCA problem there are m alternatives $(A_1, A_2, ..., A_m)$ and n criteria $(R_1, R_2, ..., R_n)$. Each alternative is appraised by those criteria (or attributes). The matrix which displays the score of each alternative based on all the criteria is denoted by $Q = (q_{ij})_{m \cdot n}$. Let $W = (w_1, w_2, ..., w_n)$ be the weight vector which satisfies the following condition: $\sum_{j=1}^{m} w_j = 1$.

The TOPSIS consists of these steps:

1. Complete the decision matrix Q and normalize it by means of either vector or linear sum normalization.
2. Weight Q by multiplying the normalized matrix by the relative importance of the attributes: $v_{ij} = q_{ij} \cdot w_j$ (being $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$), where q_{ij} represents each element of Q and v_{ij} is the weighted normalized element. The weighted Q is called V.
3. Compute the PIS and NIS by doing:

$$PIS = [v_1^+, v_2^+, ..., v_n^+] \quad where \quad v_j^+ = max_i(v_{ij}) \tag{B.1}$$

$$NIS = [v_1^-, v_2^-, ..., v_n^-] \quad where \quad v_j^+ = min_i(v_{ij}) \tag{B.2}$$

4. Calculate the distance of each alternative from PIS and NIS:

$$d_i^+ = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^+)^2} \tag{B.3}$$

$$d_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^-)^2} \tag{B.4}$$

   where $i = 1, 2, ..., m$

5. In the end, find the *closeness coefficient* [77] CC_i which is defined as:

$$CC_i = \frac{d_i^-}{d_i^+ + d_i^-} \quad where \quad i = 1, 2, ..., m \tag{B.5}$$

The best alternative is the one with the highest closeness coefficient, i.e. the one which has a final score which is the closest to the PIS.

Appendix C

Triangular grids

In Section 5.3.3 the Grid initialization by means of *triangulation* has been illustrated. Since the `PyDistMesh` tool was not available, the algorithms have all been developed by the author.
In their paper, Wang et al. [62] also investigate on the orientation of the grid. In this study, for reasons of simplicity, this will be neglected. The *equilateral-triangle grid* strategy is as follows.

Algorithm 8 Equilateral-triangle grid creator

set: desired spacing among turbines, desired number of turbines, boundaries (S), $x_{min}, y_{min}, x_{max}, y_{max}, linspace_x = 0, linspace_y = 0$
initialise $x = x_{min}$ *and* $y = y_{min}$*, number of turbines = 0*
WHILE $x < x_{max}$ **DO**:
    $x = x + spacing$, $linspace_x = linspace_x + 1$ % How many sectors to divide x-axis into
END
WHILE $y < y_{max}$ **DO**:
    $y = y + \frac{\sqrt{3}}{2} * spacing$, $linspace_y = linspace_y + 1$ % How many rows. The rows are $\frac{\sqrt{3}}{2} * spacing$ from each other - equilateral triangle
END
set allowable x, y *positions:* $x = linspace(x, x_{end}, linspace_x), y = linspace(y, y_{end}, linspace_y)$ % In this way we create a rectangular grid, being the x points far from each other by exactly the desired space and being the y points far $(\frac{\sqrt{3}}{2} * spacing)$ from each other
FOR *all the rows* **DO**:
    IF *the row is odd* **DO**:
        do not modify the row
    ELSE:
        shift all the points in the row by $\frac{spacing}{2}$ % in this way every two rows we shift to the right all the available points, to create the equilateral-triangle grid
    END
END
WHILE *number of turbines ¡ desired number of turbines* **DO**:
    WHILE *turbine is not within the boundaries* **DO**:
        randomly place a turbine in the equilateral-triangle grid
        number of turbines = number of turbines + 1
    END
END

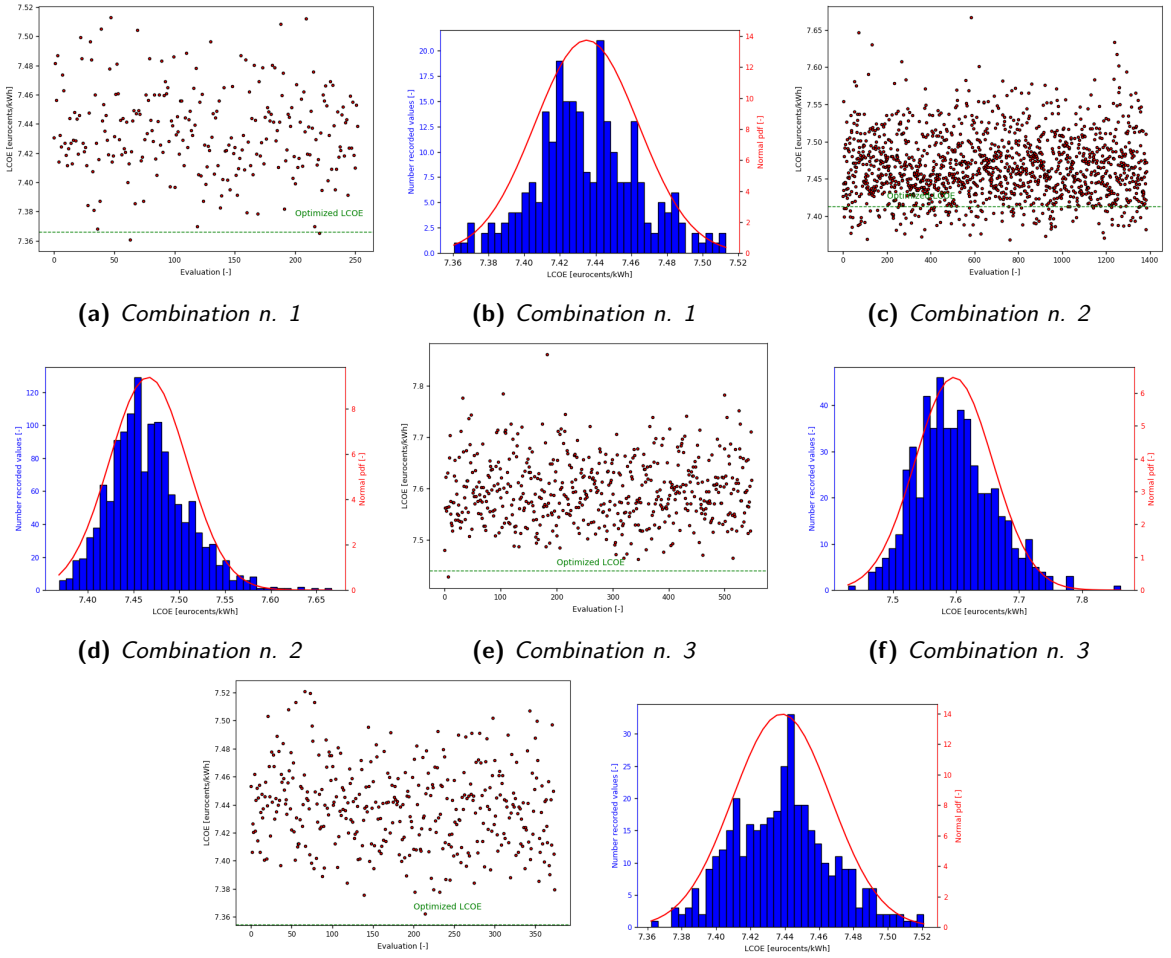
Algorithm 9 Random trinagled Initialization

set: spacing, boundaries (S), desired number of turbines
initialise $N_{spots} \neq 0, N_{turbines} = 0$
WHILE $N_{spots} \neq 0$ **DO**:
    WHILE *spot does not violate the constraint* **DO**:
        IF *iterations in while loop = 3000* **DO**:
            stop while loop
        ELSE DO:
            add spot, $N_{spots} = N_{spots} + 1$
    END
END % Now we have the maximum number of points the space which can be put in the search space without violating the constraints.
Perform Delaunay triangulation to create a meshgrid with the computed spots[1]
WHILE $N_{turbines} <$ *desired number of turbines* **DO**:
  place a turbine in one of the spots computed before
END % the turbines can occupy these spots only

Appendix D

Random sampling: graphs

These figures are taken from the case study of Eneco Luchterduinen

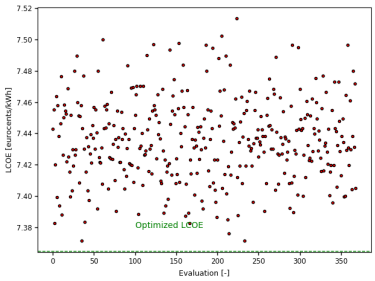


(a) *Combination n. 1* **(b)** *Combination n. 1* **(c)** *Combination n. 2*

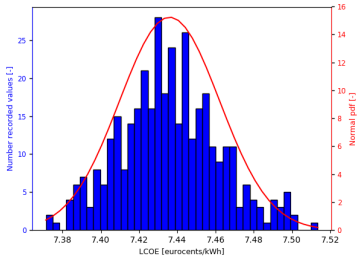
(d) *Combination n. 2* **(e)** *Combination n. 3* **(f)** *Combination n. 3*

(g) *Combination n. 4* **(h)** *Combination n. 4*

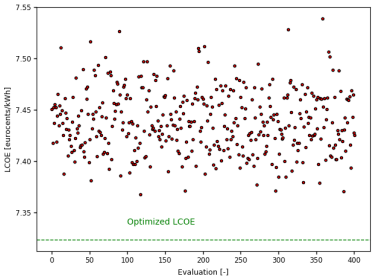
Figure D.1: *First 4 combinations - random sampling*



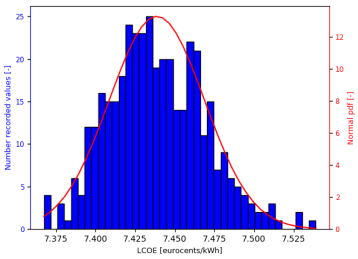
(a) *Combination n. 5*



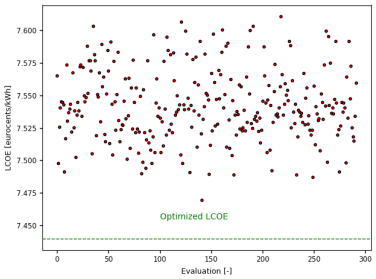
(b) *Combination n. 5*



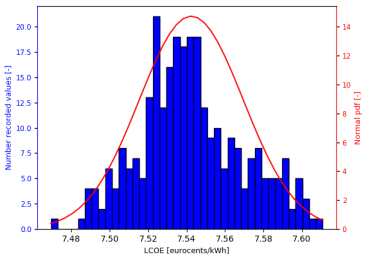
(c) *Combination n. 6*



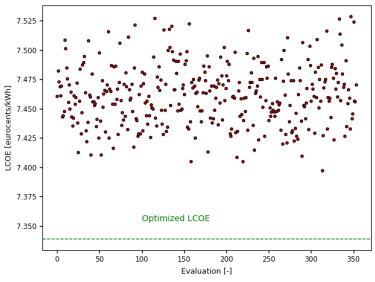
(d) *Combination n. 6*



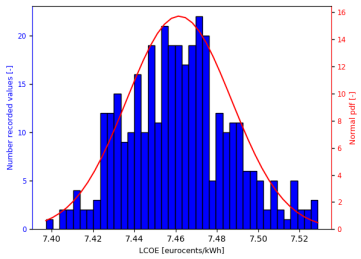
(e) *Combination n. 7*



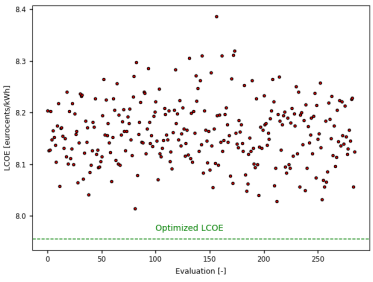
(f) *Combination n. 7*



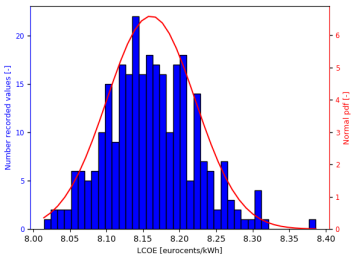
(g) *Combination n. 8*



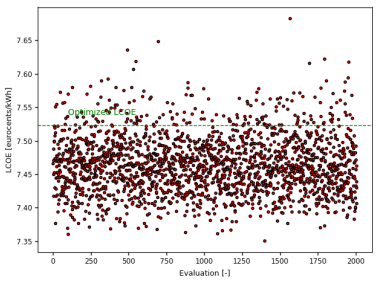
(h) *Combination n. 8*



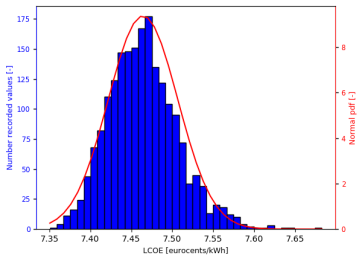
(i) *Combination n. 9*



(j) *Combination n. 9*

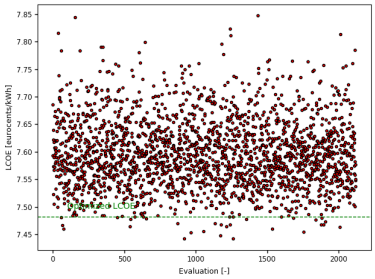


(k) *Combination n. 10*

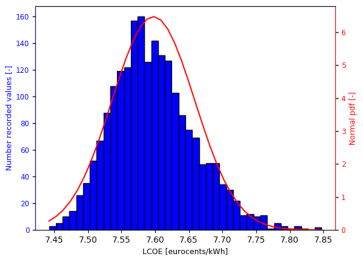


(l) *Combination n. 10*

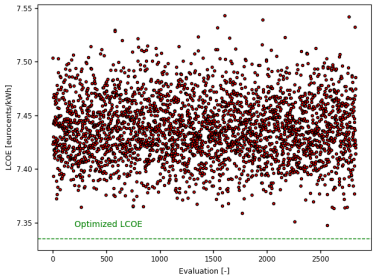
Figure D.2: *Combinations 5-10 - random sampling*



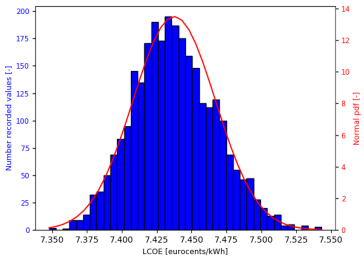
(a) *Combination n. 11*



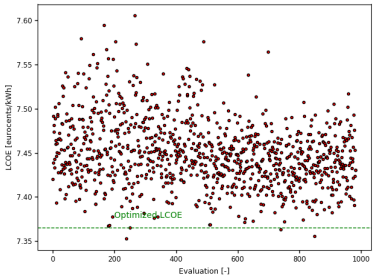
(b) *Combination n. 11*



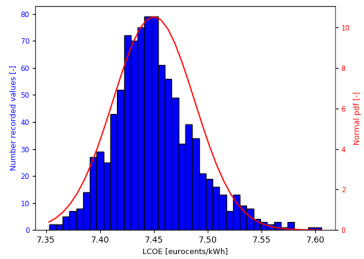
(c) *Combination n. 15*



(d) *Combination n. 15*

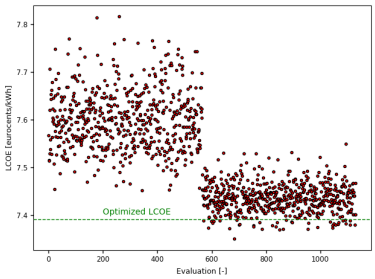


(e) *Combination n. 16*

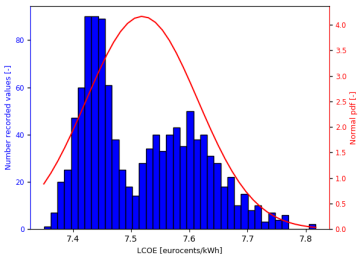


(f) *Combination n. 16*

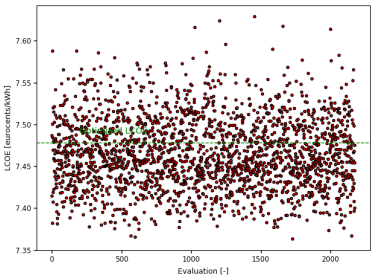
Figure D.3: *Combinations 11-16 - random sampling*



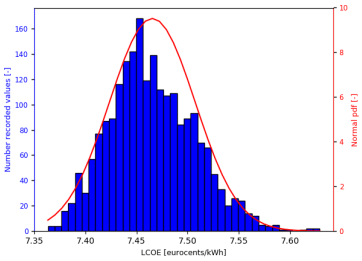
(a) *Combination n. 17*



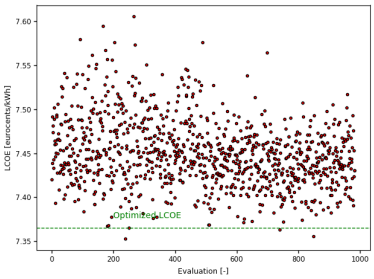
(b) *Combination n. 17*



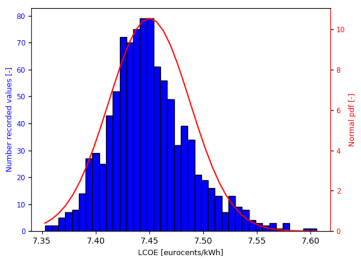
(c) *Combination n. 18*



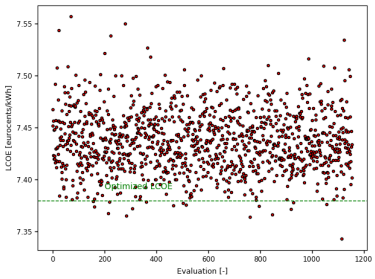
(d) *Combination n. 18*



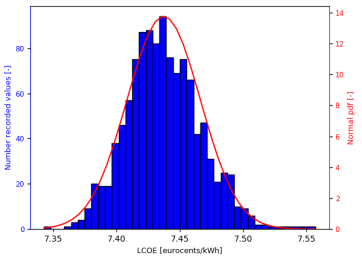
(e) *Combination n. 19*



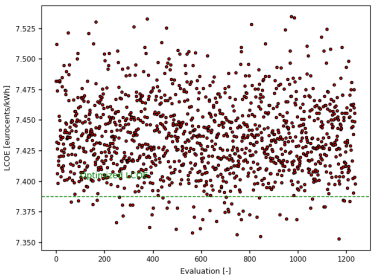
(f) *Combination n. 19*



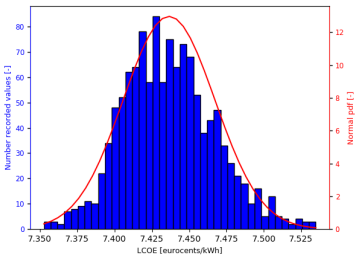
(g) *Combination n. 20*



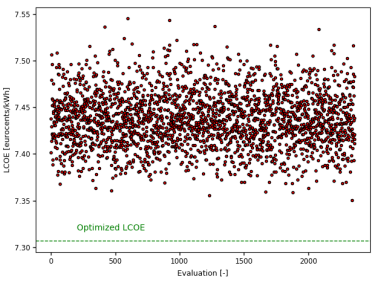
(h) *Combination n. 20*



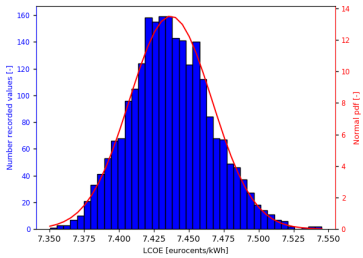
(i) *Combination n. 21*



(j) *Combination n. 21*



(k) *Combination n. 22*



(l) *Combination n. 22*

Figure D.4: *Last combinations - random sampling*