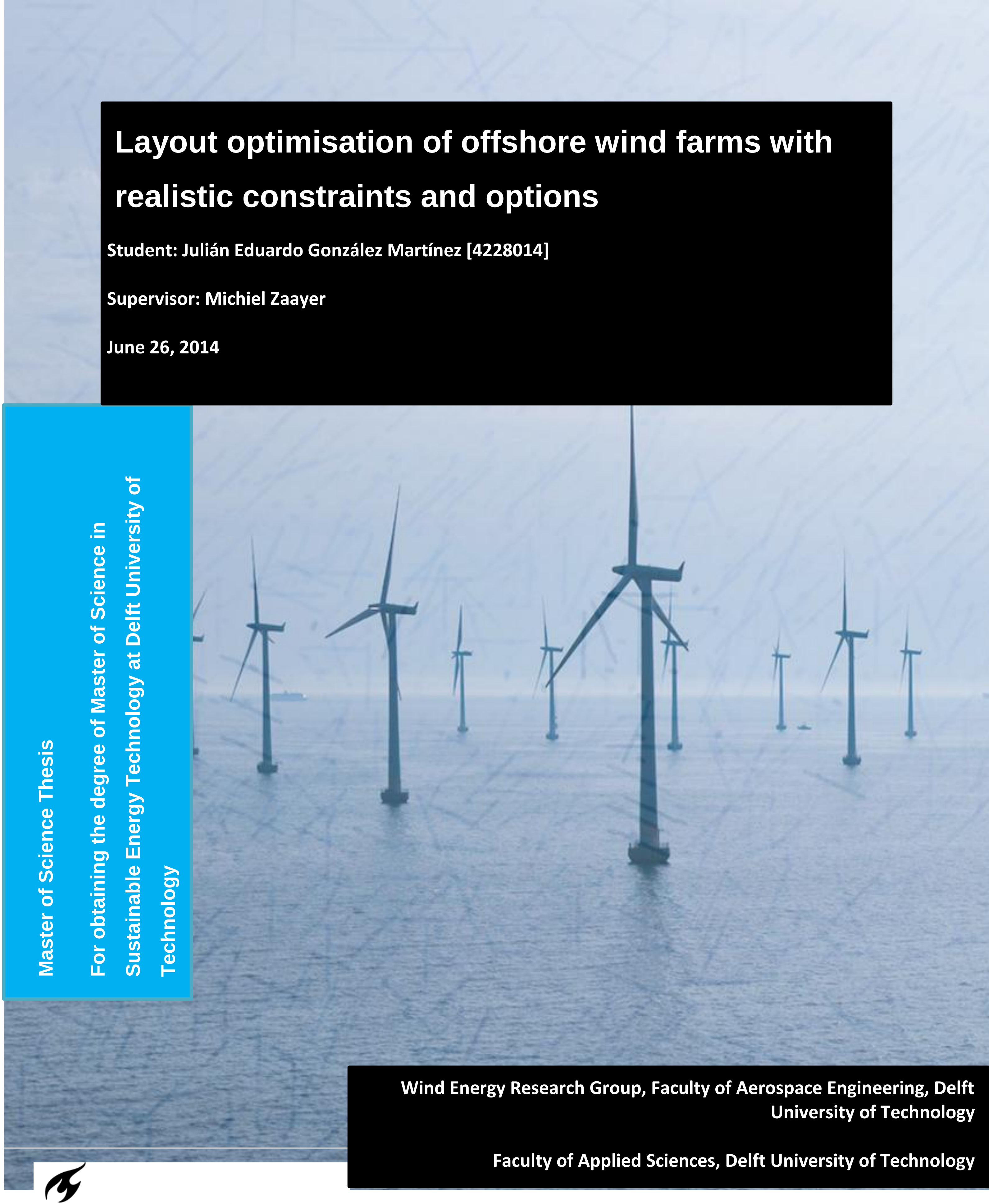


Layout optimisation of offshore wind farms with realistic constraints and options

Student: Julián Eduardo González Martínez [4228014]

Supervisor: Michiel Zaayer

June 26, 2014

Master of Science Thesis

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

Wind Energy Research Group, Faculty of Aerospace Engineering, Delft University of Technology

Faculty of Applied Sciences, Delft University of Technology

TU Delft Delft University of Technology

Challenge the Future

*“To be successful you have to have your heart in your business
and your business in your heart”*

-Thomas J. Watson

Acknowledgements

Dedicated to my parents, for their continuous support throughout all these years that enabled me to become the man I am today.

Special thanks to Michiel Zaayer and Silvio Rodrigues for providing their guidance, advice and support for the development of this project. This is yours as much as it is mine.

Contents

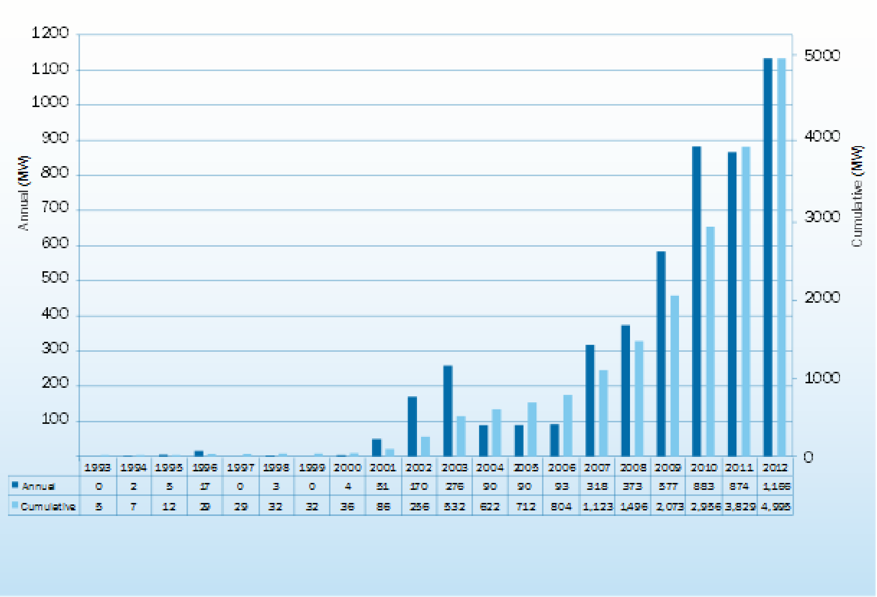
List of Figures

List of Tables

1. Introduction and Background

1.1. Motivation

The offshore wind power industry has experienced a very rapid growth in the past decade, especially in Europe, where installed capacity is increasing every year. Just in 2012 a total of 1166 MW were deployed in the European Union, which translates to 293 wind turbines distributed in 9 new wind farms. The investments made (between €3.4 bn to €4.6 bn) represent an increase of 33% in deployed capacity with respect to 2011. In total, 4995 MW of offshore wind power has been installed in the EU so far, and this number is expected to keep on rising in the foreseeable future (1).



	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
■ Annual	0	2	5	17	0	3	0	4	51	170	276	90	90	93	318	373	577	883	874	1,166
■ Cumulative	5	7	12	29	29	32	32	36	86	256	532	622	712	804	1,123	1496	2,073	2,956	3,829	4,995

Country	UK	DK	BE	DE	NL	SE	FI	IE	NO	PT	Total
No. of farms	20	12	2	6	4	6	2	1	1	1	55
No. of turbines	870	416	91	68	124	75	9	7	1	1	1,662
Capacity installed (MW)	2,947.9	921	379.5	280.3	246.8	163.7	26.3	25.2	2.3	2	4,995

Figure 1. Annual and cumulative offshore wind installed capacity in the EU, from 1993 to 2012. Reproduced from (1).

With the number of offshore installations growing it is in the best interest of researchers, developers and the general public to maximize the net yield of wind farms, which would make them more efficient while driving down the costs of such massive projects. Heavy emphasis has been placed on optimization of offshore wind farm layouts, mainly to drive down the costs of the energy produced (Levelized Production Cost, LPC) in order to make projects more attractive for investment. Modeling tools have been extensively developed in order to gain a better understanding of intra-array dynamics and optimize characteristic farm parameters such as yield, losses, costs, turbine distribution, etc.

Recently heavy emphasis has been placed in research and development of automated wind farm design and optimization tools, which are capable of determining the optimal position for each of the turbines within a project area based on site conditions, interactions between the turbine wakes and additional constraints that reflect in the final annual energy yield and levelized production cost (LPC) of energy.

Several of these developed tools and projects have attempted to implement advanced algorithms for automated offshore wind farm design based on objective function optimization. Usually LPC or energy yield are chosen as objective functions; most optimization routines (from single objective optimization to genetic algorithms) account for wake effects, turbine characteristics, water depth, distance to shore and site conditions to arrive to a solution. However, other design considerations like aesthetics, practicality, noise effects, environmental interactions or other “hard to account for” variables are rarely considered in optimization models and must therefore be manually adjusted by the designer moving away from the optimum solution even further than required. Therefore, it would be nice to explore the possibility of accounting for these kind of not-so-common objective functions in pre-developed OWF design software in order to apply some more real world constraints into an otherwise exclusively cost-driven optimization routine.

1.2. Objectives

The objective of the project is to design, develop refine and implement a series of modifications and extensions in the form of algorithms that enable an existing offshore wind farm design tool (from here on referred to as the MZ Tool) to develop wind farms with complex layouts subject to realistic constraints. Therefore, the final production of the project is a functional software tool with extended functionality when compared to the one in existence currently.

In order to achieve this main objective four secondary goals or activities have been defined, which are:

- Investigate about the needs of wind farm developers and their current limitations when using tools similar to the one targeted by this project and translate them into functional requirements. This also covers performing a state-of-the-art analysis.
- Prioritize the functional requirements and determine which of the problems they pose are feasible to be solved within the scope of this project.

- Develop solutions in the form of algorithms and routines designed to solve the problems identified in the previous step.
- Implement and test the solutions within the MZ tool to determine the extent of success of the project.

The proposed approach closely follows the waterfall product development model as presented in Figure 2.

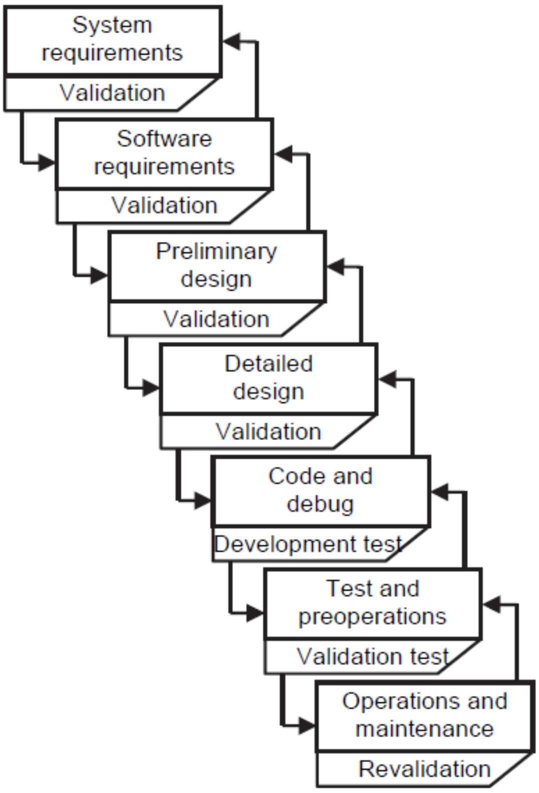


Figure 2. Waterfall model for product development. Reproduced from (2).

The operations and maintenance phase as depicted in the model is the only one that is not contemplated for the current project, although it may be argued that this entire project is an “operation and maintenance” procedure on the current prototype of the MZ tool.

1.3. Document Outline

The outline of this report is as follows:

Chapter 2 provides insight on the state-of-the art of wind farm layout optimization as well as a summary of other aspects that need consideration when developing such projects. It serves to provide the reader context on the overall problem.

Chapter 3 presents an overview of the MZ tool, pointing out is most relevant features, characteristics and limitations.

Chapter 4 defines which are the user requirements for wind farm design and optimization tools. These user requirements are then classified and prioritized.

Chapter 5 presents the process used to translate the identified requirements to be addressed into a complete problem definition.

Chapter 6 describes the process to develop a solution to the cable layout optimization, which was an issue identified as priority in Chapter 5.

Chapter 7 describes the process to develop the wind farm layout optimization algorithm following guidelines and requirements previously identified.

Chapter 8 contains a discussion on the performance and results of the implemented solutions.

Chapter 9 presents the conclusions of the project as well as recommendations and pointers for future developments.

2. Computational methods for layout optimization

2.1. Overview of optimization

Optimization problems are amongst the most recurring ones in engineering. They are concerned with finding the minima and maxima of a function, often named an objective function. In order to do this the values of the problem variables are chosen from within an allowed set (often called the solution space) until the objective function attains its maximum or minimum value. A mathematical formulation of the generalized maximization problem presented in (3):

$$Given\ f: A \to \Re$$

$$Sought: x_0 \in A\ such\ that\ f(x_o) \geq f(x)\ \forall\ x\ \ \in A$$

Equation 1.

The subspace *A* is often defined by a set of inequalities, equalities or constraints that all the elements within have to satisfy. Therefore, *A* constitutes the solution space and its elements are called feasible solutions.

Minimum and maximum points may be local or global, which further increases the complexity of the problem and can sometimes lead to undesired or incorrect solutions. The most basic optimization routines, denominated local search algorithms, start from an initial guess as an approximation to the solution and then modify and test it until no better solution is attainable. One of the most popular local search methods, known as the gradient method, is presented in Figure 3.

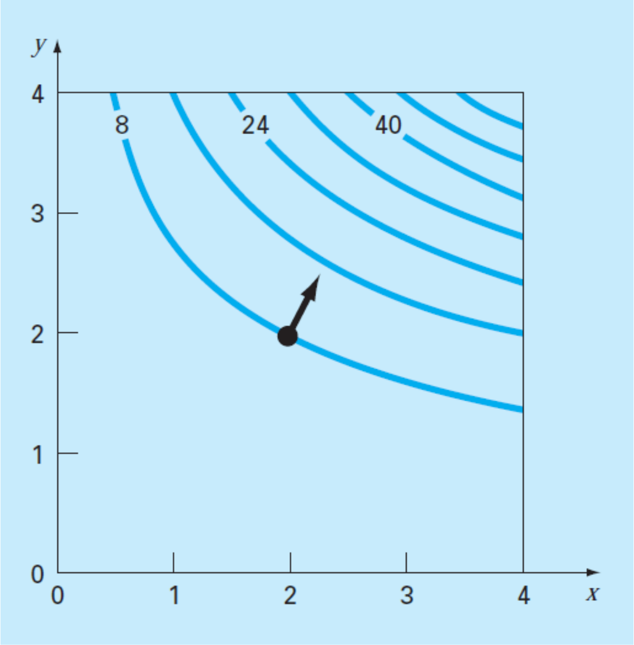


Figure 3. The gradient method, also known as steepest ascent, shown for a multidimensional unconstrained optimization problem. Reproduced from (4).

This method tries to find the optimum by “climbing” the solution space until the maximum is reached. If the black dot represents the initial guess, then it is altered by a certain magnitude (represented by the arrow), and the process is repeated until no further alteration results in a better solution. While these kinds of optimization methods work flawlessly for most simple problems they become problematic when the solution space becomes more complex, in which case local and global maxima/minima may exist within the same solution space. A 3D representation of such complex solution space is presented in Figure 4.

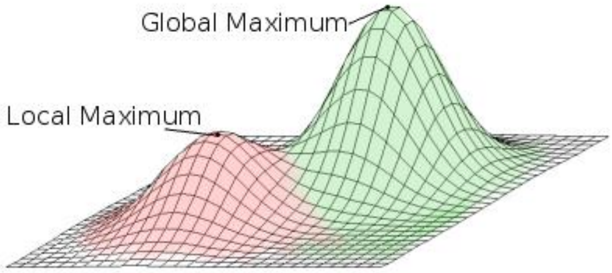


Figure 4. 3D mesh of a non-concave solution space, with local and global maxima. Reproduced from (5).

Suppose a traditional local search method is used to try to find the global maxima of the function. It becomes clear that if the initial guess is closer to the local maximum (for example, within the red meshing) the gradient method and most local search methods will end up with the local maximum as a solution. However, it is often the case that the global maximum is the desired solution. Thus, it becomes clear that local search methods are highly dependent on the initial guess. Fortunately, other optimization methods have been devised to bypass this problem. Some of these will be discussed in detail in the following section.

2.2. State of the art of layout optimization tools

The offshore wind farm design problem is characterized by several interconnected variables, which often makes it difficult to find an optimal solution without the help of computational tools and techniques. Different optimization techniques are available to solve this kind of problem; in general they can be split in two groups: exact optimization algorithms and approximation algorithms, also known as heuristics. The second category has been proven to be better suited for the layout optimization problem for reasons that shall now be discussed.

Pérez et.al (6) present two main reasons why heuristic methods are preferred to exact optimization for wind farm layout problems:

- The computational time of gradient-based methods is often prohibitive to solve these kinds of problems.
- The optimal location problem is non-convex, and gradient based methods attain local solutions as was explained in section 2.1. Depending on the initial guess the global optimum may be skipped.

Approximation algorithms "search the solution space intelligently, and focus on those sub-spaces of the total solution space which have a higher probability of finding a high quality, feasible solution" (7). Khan et.al (7) further divide approximation algorithms into two sub-categories: constructive and iterative. Constructive algorithms find a solution by using deterministic moves to alter the previous guess; they are faster than iterative algorithms but often produce solutions of unacceptable quality or can even fail to produce a solution at all. Iterative algorithms, on the other hand, attempt to find the solution by performing a controlled walk over the solution space. Iterative non-deterministic algorithms are the most widely used in the field of wind farm optimization due to the advantages hereby presented and their ability to cope with the complexity of the problem.

A literature review revealed that four iterative algorithms stand out as the most widely used for layout optimization: genetic algorithms, differential evolution, particle swarm optimization and simulated annealing. Nevertheless, additional approaches exist. The most important types of algorithms will now be described in detail.

2.2.1. Genetic algorithms

Inspired by the process of natural selection, these algorithms have proved to be highly effective in solving the wind farm optimization problem. Particular solutions in a pool are regarded as chromosomes and only the fittest ones will survive and be able to reproduce, that is, effectively transmitting their characteristics (called genes) to their offspring. After two chromosomes are allowed to reproduce a new set of chromosomes, called offspring, are produced. Offspring are generated by means of four genetic operators which mimic the biologic process of genetic recombination: selection, crossover, mutation and inversion. The process stops once a certain stopping condition is met and the final chromosome is returned as the solution.

A seminal paper by Mosetti et.al (8) is regarded as the first attempt at solving the layout problem via a genetic algorithm. Written in 1994, this paper uses a multi-objective optimization approach, with the goal of maximizing energy output and minimizing cost of the wind farm. The solution is given as a string of chromosomes that represent turbine positions in a rectangular grid. Grady et.al (9) tackles the problem in a similar fashion to Mosetti more than a decade later, with the main difference being a change in the objective function: he uses a single objective optimization approach, with the objective function being the ratio of cost to power output.

Attempts have also been made by several authors to give more degrees of freedom to the solution space of the genetic algorithm, in order to find more realistic solutions to the problem. For example, Mora et.al (10) and Huang (11) not only changed the objective function once again (to Net Present Value NPV and profit, respectively) but also allowed the number of turbines to be placed to be a variable instead of an input, thus ending with a solution that not only contains turbine positions but also the optimal number of turbines for the wind farm. Huang later modified his original genetic algorithm (12) and transformed it into a so-called single objective hybrid algorithm, in which both local and global search are performed simultaneously, drawing from the advantages of both approaches.

Two papers, written by Sisbot et.al (13) and González et.al (14) provide an interesting approach to the problem due to their attempts at further constraining the placement problem with forbidden positions that reduce the solution space to reflect a real life scenario. Sisbot uses a multi-objective optimization approach for a real site located in the Aegean Sea. The objective of this project was to design a wind farm subjected to particular site conditions and constraints of the terrain which was accomplished by restricting the available siting positions for a wind turbine in the grid that constitutes the solution space. González's approach is based on the problem definition by Grady et.al (9) for an onshore wind farm. However, González added the presence of a main road crossing the wind farm, two areas where turbine placement is forbidden and a load-bearing capacity zone in which foundation costs for turbines are higher. A more realistic NPV cost model was also introduced as well as a more complicated wake model when compared to the landmark one developed by Jensen (15). In posterior works González et.al (16) (17) further refined their optimization approach by introducing uncertainties in the annual energy production due to the unpredictable nature of wind. Also, other variables that in reality have an associated uncertainty, namely pricing of energy and the discount rate were modified to reflect such sensitivities.

One final implementation of a genetic algorithm with a novel approach is presented by Kwong et.al (18), who for the first time introduced an environmental consideration in the objective function in the form of the noise generated by the wind farm. In their multi-objective approach, they sought to maximize power generation while minimizing noise levels at the wind farm boundary.

Genetic algorithms are probably the most widely used iterative non-deterministic optimization methods to solve the optimal layout problem. Further works on the subject using this approach are presented by Wan et.al (19), Wang et.al (20) (21), Herbert-Acero et.al (22), Emami et.al (23), Mittal (24), Kusiak et.al (25), Bilbao et.al (26), and Khan et.al (27).

2.2.2. Differential Evolution

In a similar way to genetic algorithms, differential evolution uses an approach based on natural phenomena to produce solutions. The genetic operators previously mentioned (mutation, crossover and selection) are also used, although in a different manner. These algorithms maintain two arrays: one which contains the current population of individuals, while another one stores those which will be passed on to the next generation. In order to determine which individuals are the fittest ones competition is simulated between the existing arrays and some trial ones, until the strongest solution is achieved.

As Khan Et.al (7) point out, application of differential evolution algorithms to wind farm problems has been limited so far, with only two papers written by the same authors actively working on the subject. The work by these authors, namely Rasuo et.al (28) (29) is not explored here in detail.

2.2.3. Particle swarm optimization

Particle swarm optimization algorithms, often shortened to PSO, are algorithms which use a population of individuals to find an optimum solution by making each individual move about the search space and “remember” which its best position is so far. In the PSO context individuals are regarded as particles, with each particle being a potential solution to the problem. The collection of possible solutions is regarded as the swarm. Tesauro et.al (30) explain a common misconception of the PSO approached applied to wind farm design and also manage to synthesize the inner workings of the method: “Thinking of the wind farm layouts as particles of the swarm is quite difficult because we naturally tend to associate the turbines to particles and the wind farm to the swarm, which is wrong. In reality the whole wind farm layout is a bird/fish that is moved around by the algorithm, where “moved around” means that the position of the wind generators within the layout is changed”.

In order for a particle to move it has an associated velocity to search the solution space in a controlled manner. Particles can also communicate with one another and trade information regarding the search space, thus guaranteeing that the final solution will be the global minimum/maximum. It should be noted that PSO algorithms are highly parameter dependent, which is why proper model tuning and calibration is required.

The first attempt to solve the wind farm layout problem using a PSO approach was by Rahmani et.al. (31), employing a single objective optimization and comparing the results with those obtained by other authors using genetic algorithms. Wan et.al (32) further worked on the PSO technique by adding parameters such as inertia weight, maximum velocity and a constriction factor. However, his most interesting contribution was using a continuous solution space instead of a discrete one as all approaches to the problem previously have, effectively making it possible for any wind turbine to sit in any position inside the wind farm area. Wan’s approach was also single-objective optimization.

Additional works using PSO approaches are those of Chowdhury (33) (34) et.al and Bilbao et.al (35).

2.2.4. Simulated Annealing

Unlike genetic algorithms, differential evolution and particle swarm optimization, simulated annealing (SA) algorithms are not population based; instead they work with only a single solution which is refined until no further improvement on it is achievable. The algorithm has two main steps: the initialization and a so-called Metropolis procedure. The former one is concerned with generating a feasible solution, while the latter one perturbs it to generate a secondary solution. If the quality of the secondary solution (usually called fitness) is higher than that of the original, the secondary solution replaces its forerunner as the best solution so far. While this method is quite similar to a local search algorithm, SA uses an additional probabilistic device to avoid getting “stuck” with local minima/maxima. If the fitness of the new solution is not greater than that of the previous one then it is not readily discarded;

instead it may be accepted based on the Metropolis criterion, which is described by the following equation:

$$P(r < e^{-\frac{\Delta h}{T}})$$

Equation 2.

Where *r* is a random number ranging from 0 to 1, *Δh* is the fitness difference between solutions and *T* is a parameter known as the annealing temperature. This is due to the fact that SA was originally formulated for the problem of cooling a material at high temperature in a heat bath. The reason behind the use of the Metropolis criterion allowing the acceptance of not necessarily better solutions than the previous one is best described by Aghabi (3): he describes the overall optimization process as finding a dip or a peak in the function, corresponding to minima and maxima, respectively. If a minimization is desired then a local search algorithm will only perform “downhill” moves until a minimum is reached, be it local or global. However, an SA approach allows occasional “uphill” moves that prevent the routine from getting trapped in the neighborhoods of a local minimum.

As is the case for PSO, SA algorithms are also highly dependent on proper parameter tuning, being the most important the initial temperature, the cooling ratio, the stopping temperature and the length of the Markov chain, which determines the time used until the next update. The work by Aghabi (3) is a comprehensive implementation of a SA algorithm to solve the layout optimization problem, which also provides pointers to aid with the proper selection of the parameters previously mentioned.

2.2.5. Evolutionary Strategies

In principle, evolutionary strategies (ES) seem quite similar to genetic algorithms. Both types are biologically inspired routines that aim to replicate processes such as mutation, recombination and parent-offspring dynamics within single and multiple objective optimization problems. What marks the difference between evolutionary and genetic strategies is that in ES new candidate solutions are sampled stochastically from a distribution, instead of the arbitrary recombination process used in genetic algorithms.

The Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) represents the most recent generation of ES algorithms, offering a number of significant performance advantages when compared to other heuristic approaches and has been applied to a wide range of engineering problems with success. An example of the CMA-ES used to solve the layout optimization problem is presented in the work of Rodrigues et.al (36).

2.2.6. Other approaches

Sections 2.2.1 to 2.2.5 described the most widely used algorithms and approaches at solving the layout optimization problem, however, there are some more that although aren’t quite as popular are still worth mentioning.

Viral algorithms have inner workings similar to genetic algorithms, although their main inspiration was the mechanisms that viruses use to replicate and transmit themselves. The process starts with an initial population of possible solutions, the most fit of them are replicated and their traits propagated according to the lytic cycle, which govern viral reproduction in reality. The process is as follows: “the virus injects its own nucleic acids into a host cell, which mistakenly copies the viral acids instead of its own. In the next step the viral DNA forms new viruses inside the cell which then break the cell membrane to infect other cells” (30). The solutions with lowest fitness values are grouped in a subpopulation which is recombined to achieve higher fitness, after which they can form part of the main cycle.

Greedy heuristic algorithms start out with an initial guess for a solution. Then, three operations can be performed on the layout: adding a turbine, removing a turbine or moving a turbine. The altered layout is tested and if it has a higher performance to the previous one it replaces it as the new best solution. To avoid local minima and maxima turbines can be occasionally moved a random distance. The process is stopped when continued iterations no longer produce better results.

Interestingly, some researchers have managed to apply pure or modified local search algorithms for the layout optimization problem, such as Wagner et.al (37). However, their approach seems to borrow heavily from greedy heuristic algorithms, which would classify it as a hybrid approach. The industry tool AWS OpenWind uses a local search optimization (38).

In addition to Wagner et.al, other authors have also produced hybrid approaches which aim to retain the advantages of their preceding algorithms while diminishing the effect of their disadvantages. Saavedra-Moreno et.al (39) use a greedy heuristic algorithm to generate an initial solution (seed) which then serves as an input to an evolutionary algorithm, which results in an improvement in convergence times and solution quality by guaranteeing that the initial guess is a good one. Another notable hybrid approach is the work by Pérez et.al (6); they use a heuristic algorithm to generate an initial solution which is then optimized by means of non-linear mathematical programming for local search. Both approaches strive to apply usable area constraints to the solutions, limiting the valid positions for turbine placement.

2.3. Objective functions

The approach selection plays a vital role in the solution of the problem; however, the optimization process may produce undesired or non-useful results if an adequate objective function is not used. The most popular choices for objective function are:

- Energy yield: this function is also known and referred to as annual energy production (AEP) in a great deal of literature is the most straightforward objective function to maximize; the optimization strives to obtain the layout that produces the highest amount of energy. AEP is often used in commercial software as the objective function, such as GL Garrad Hassan WindFarmer, AWS OpenWind and EMD WindPro.
- Cost of energy: this function considers both energy and cost parameters, which makes it a more robust choice when compared to AEP. It is simply defined as the cost of energy per kWh produced. The cost of energy can be defined in different ways, the simplest approach being to take the ratio between total project costs over energy yield (E_y). However, levelized production cost (LPC) is a better indicator of the true cost of energy, since it not only considers investment costs (C_{inv}) but also operation and maintenance costs ($C_{O\&M}$), decommissioning costs (C_{decom}) and an annuity factor (a) to account for the change in value of money over time. LPC is often defined as:

$$LPC = \frac{C_{inv}}{a * E_y} + \frac{C_{O\&M}}{E_y} + \frac{C_{decom}(1+r)^{-T}}{a * E_y}$$

Equation 3.

- Profit: the profit function is an economic indicator of the performance of the wind farm, as the objective is to maximize net gains resulting from the operation of the turbine array over the course of its lifetime. According to the review by Tesauro et.al (30), “Rethoré et.al (40) built the most comprehensive function within the TopFarm framework”. The function they used, named financial balance, is defined as follows:

$$FB = P_e * TEP - C_D - C_M - \left(C_f + C_g\right) * (1 + \left(\frac{r_l - r_i}{N_L}\right))^{X*N_L}$$

Equation 4.

P_e is the price of energy in the market, TEP is the total energy production across the lifetime of the project (X), C_D is the degradation cost of the turbines, C_M are the maintenance costs, C_f is the future value of the investment in foundations and C_g the same parameter but for electrical grid investments. The inflation and interest rates (r_i

and r_l, respectively) also play a part in the equation. The final parameter, N_L, represents the number of times interests of loans have to be paid per year.

- Net Present Value (NPV): a direct alternative to using Profit, the NPV indicates the financial performance of the wind farm over its lifetime using a different, well known formulation (not presented here).

Combinations of the objective functions are also a possibility to create custom-built optimization targets, as many authors have suggested. One such example, albeit a very famous one, is the proposed objective function by Mosetti et.al (8), structured as follows:

$$Obj = \omega_1 * \frac{1}{P_y} + \omega_2 * \frac{C_{tot}}{P_y}$$

Equation 5.

where P_y is the yearly power produced, C_{tot} is the cost of the project and ω_1, ω_2 are two weighing constants to be chosen. If ω_1 is set to 1 and ω_2 to 0 the objective function targets maximization of energy yield. Likewise, if ω_1 is set to 0 and ω_2 to 1 then the minimum cost of energy becomes the objective.

2.4. Additional problem considerations and models

Although the optimization routine for the layout may be considered the centerpiece to the problem there is a great number of parallel aspects akin to wind farm design that need to be considered because they either affect or are affected by changes in the layout. The most important of them are presented here.

2.4.1. Wake modeling

The study of wind turbine wakes is not a new subject. When a rotor disc interacts with the incoming flow a pressure drop is generated at the site of the interaction, which is essential in converting the kinetic energy of the wind into output power via aerodynamic power conversion. Nevertheless, this interaction also has an effect on the incoming wind field; since some of the energy is extracted from the flow there is a velocity dip (from here onwards called the deficit) that is a function of the downwind distance from the wind turbine. Other turbines places downwind of said turbine will therefore be in the wake of the upwind one, and will be subjected to different wind characteristics than those of the undisturbed flow.

There is a great deal of wake models available, nevertheless a quite simple model by N.O Jensen (15) and then modified by Kátic et.al (41) remains, up until now, the most widely used of them all. Although simple in nature, explicit models such as the Jensen model are still widely used due to their ease of implementation and low computational demands, while still providing satisfactory results. The details of the model are not presented here, but may be found in their original references or in (42).

Out of all the authors that have attempted to solve the wind farm layout problem presented in section 2.2 only Rasuo et.al (28) (29), González et.al (14) and Chowdbury et.al (33) use alternate models to the Jensen/Kátic approach. The former developed their own wake model, while the two latter authors used the Frandsen wake model (not discussed here).

2.4.2. Cost models

Cost modeling is a centerpiece behind proper definition of the objective function for the optimization process. Many of the objective functions presented in section 2.3 have an included cost term (that can account for construction, operation and maintenance and decommissioning phases); however there is no common consensus of how said total costs are best estimated. Some authors assemble their own cost functions, while others resort to guidelines and estimations based on previous data.

Khan et.al (7) presents percentage approximations to the costs of a wind farm based on several studies published before. Table 1 presents his proposed cost composition. Notice that although he mentions maintenance costs in his work, he provides no percent approximation to them as with the other concepts given.

Table 1. Estimation of main costs for an offshore wind farm, as reported by Khan et.al (7).

Concept	Percentage of total cost (%)
Wind turbines	65-75
Electrical infrastructure	10-15
Civil works	5-10
Installation and miscellaneous	10

Elkinton et.al (43), as part of the widely recognized OWFLO project, present a more detailed cost breakdown from two different sources, which are in turn also based on state-of-the art and case studies. The first of these studies was the Opti-OWECS (Optimization of Bottom Mounted Offshore Wind Energy Converters) while the second one was the OWECOP (Offshore Wind Energy – Cost and Potential) study. Their findings are presented in Table 2.

Table 2. Breakdown of main costs for Offshore wind farm projects, based on Opti-OWECS and OWECOP studies. Reproduced from (43).

Component	% of energy cost (Opti-OWECS)	% of Installed Cost (OWECOP)
Turbines and tower	34	25
Sub-structure and foundation	24	11
O&M	23	17
Electrical interconnection	15	17
Installation and decommissioning	Included in above	18
Other	4	12

Aside from the percentage approximation approach some authors propose using mathematical formulations that approximate the cost as a function of other problem variables. The most recurrent one of these was first introduced by Mosetti et.al (8), and has been also used by Grady et.al (9), Marmidis et.al (44), Saavedra et.al (39) and Mittal (24). The equation that describes the total cost per year of the entire wind farm is:

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

Equation 6.

where N is the number of wind turbines in the wind farm. There is no clear advantage between using one cost model approach over another, which means that it is a matter of choice for the designer. The use of these cost models becomes relevant in layout optimization only when the number of wind turbines in the wind farm is not fixed but an optimization variable.

2.4.3. Area limitations and symmetry

A problem that is intertwined with most negative issues is the area selection problem. Many of the annoyances and negative impacts of offshore wind farms can be avoided if from the planning stage a series of no-go areas are defined, either by the project leasers or the planners themselves. For example, say the project developer receives a permit from the local government to build a wind farm within a certain area *A* within their continental sea. However, within area A the developer determines that a subarea A_1 has a water depth that becomes economically prohibitive (support structures too expensive), a subarea A_2 which is known to be a common spot for recreational fishing, a subarea A_3 that has high porpoise activity, subarea A_4 was found to have an unsuitable soil type for the planned support structure and subarea A_5 is too close to the shore for the local people to give the project their support (or, as Snyder et.al (45) put it, may spoil their view of historical areas). It would be wise for the developer then to plan its wind farm in the space given by the remaining area, if it is still technically and economically feasible. This same approach can be used to deal with many constraints to the project using a single methodology, from here on referred to as area restriction.

Pérez et.al (6) present a rather simple formulation of the area restriction problem. In their work all the wind turbines must be allocated within a certain area, defined by four nodes (quadrilateral). Given the four node coordinates as (x_i^L, y_i^L);i=1,2,3,4 as input data, the constraints to ensure that all turbine remain within the polygon formed by these nodes is:

$$\begin{vmatrix} x_i & y_i & 1 \\ x_j{}^L & y_j{}^L & 1 \\ x_k{}^L & y_K{}^L & 1 \end{vmatrix} \geq 0; \begin{Bmatrix} \forall\, i; j = 1; k = 2 \\ \forall\, i; j = 2; k = 3 \\ \forall\, i; j = 3; k = 4 \\ \forall\, i; j = 4; k = 1 \end{Bmatrix}$$

Equation 7.

The constraint expressed by Equation 7 "requires that the area of the four triangles formed by each node-turbine with each pair of consecutive boundary nodes (measured in counter-clockwise direction) is always greater than zero" (6). While this formulation is to ensure that all wind turbines are placed within the boundaries given by the nodes, the conditional can be easily reversed to create areas in which turbines cannot be places (which do not necessarily have to be rectangular). Wagner et.al (37) also formulate an area restriction for their optimization algorithm, although in a much simpler manner by placing an upper bound on the available area for turbine placement.

Another common constraint often found in optimization algorithms for layouts is a minimum allowable distance between turbines, which has a perfectly reasonable physical explanation. The industry standard is usually taken as 8 to 10 turbine diameters (8D-10D) of separation between turbines to achieve array losses below 10% (46), with 4D-5D being often considered as the minimum separation before wake effects become extremely detrimental on downwind turbine performance (some issues with this statement will be treated in section 5.4). Safety reasons also play an important role in this minimum distance definition. Pérez et al (6) formulate this constraint using the following expression (using 4D as minimum distance):

$$\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{(4D)^2} \geq 1; \forall i = 1, \ldots, (N_T - 1); \forall j > 1$$

Equation 8.

Regarding public preferences for overall layouts no information is readily available in scientific literature. However, there is a strong tendency for people to prefer symmetry to asymmetry as an indicator for beauty and overall aesthetics. In 1933, a mathematician by the name of George Birkhoff proposed a mathematical formulation to determine the aesthetic value (aesthetic measure) of an object, definition which is mainly applied to art. However, his basic formulation may still be applicable for the case of an offshore wind farm. Birkhoff (47) stated that the aesthetic experience is composed of three phases: “i) a preliminary effort of attention, which is necessary for the act of perception, and that increases proportionally to the complexity (C) of the object; ii) the feeling of value or aesthetic measure (M) that rewards this effort and iii) the verification that the object is characterized by certain harmony, symmetry or order (O), which seems to be necessary for the aesthetic effect” (48). The aesthetic measure is defined as the ratio between order (symmetry) and complexity. Higher values of M indicate a superior aesthetic experience. In simple terms, Birkhoff concludes that beauty increases with symmetry and decreases with complexity. This result may be extended to wind farm arrays.

2.4.4. Electrical infrastructure

The layout choice has a deep impact on the electrical infrastructure chosen for the project, in particular the cable topology. Figure 5 presents the standard types of topologies used for power collection and transmission from the wind turbines to the transformer stations and then to the shore connection.

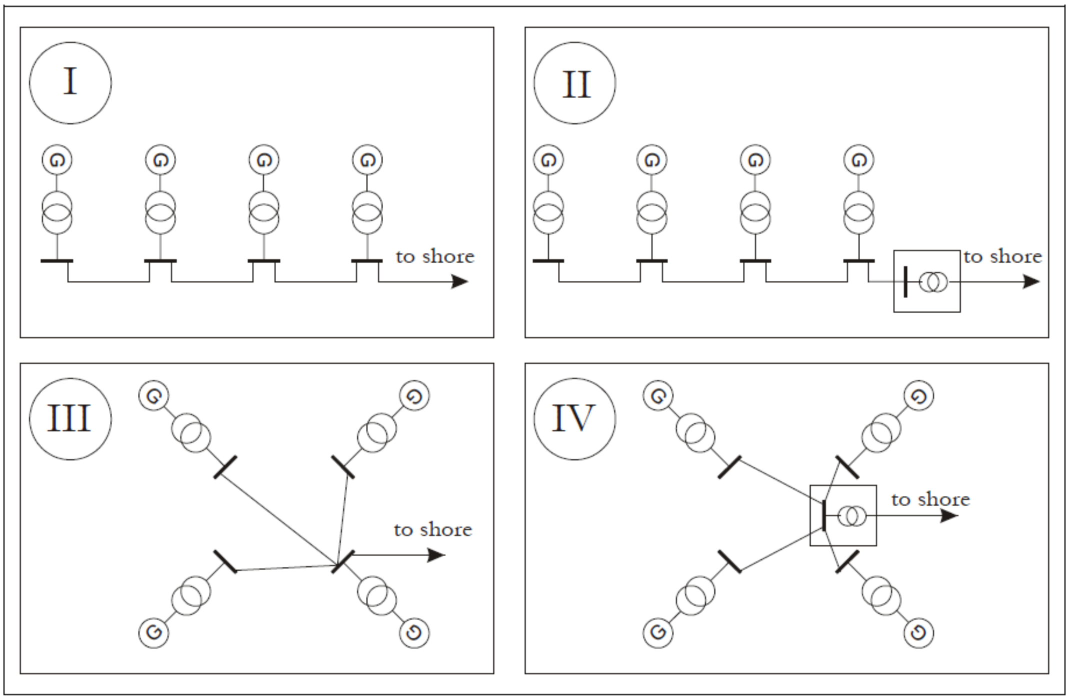


Figure 5. Star and string layout options for turbine interconnections. Reproduced from (49).

The two most common unitary topologies are shown by I and III in Figure 5, respectively named string and star. An additional transformation station may be added to allow for high voltage shore connections, as depicted in II and IV of the same figure. Schachner (49) mentions that for large offshore wind farms a combination of these unitary topologies is used, and several strings of turbines are connected to the onshore grid hookup. This has an advantage over using an strictly star layout: shorter cabling lengths and a simpler pattern; however, the layout is more sensitive to cabling failure since using a series connection means that all upwind turbines in the same string to the failure point will remain non-operational until a fix has been made.

It makes sense that once the layout is altered the overall cable topology is also affected. Often tradeoffs between variables that the designer wishes to optimize appear, and the cabling layout is one such variable that is highly sensitive to changes made to optimize other aspects of the problem. For example, moving the wind farm further away from shore may reduce the visual impact of the project, but it will also raise its overall cost by generating the need for a longer cable to connect the array to the grid onshore. Also, changes in cabling distance produces changes in the overall transmission losses which are an important variable to determine overall wind farm efficiency. Another common tradeoff occurs when spacing between turbines is changed: greater spacing may be desirable since wake losses are reduced by spreading the turbines further apart, but the length (and costs) of internal cabling increases.

In the optimization routine cabling is often considered in two ways. The first one is to include cable costs directly into the cost model as a function of total cabling distance required. This way using more cable will have an indirect negative effect on the objective function, and thus it will be desirable to minimize it. The second way cabling is considered in the routine is via a loss model for said components, which scale also with the total transmission distance. Energy output is therefore sensitive to cable topology as well, which means that in turn it has a huge repercussion on overall wind farm performance.

2.4.5. Support structures

Although not highly dependent on wind farm layout, the support structure is important to be considered because it is intertwined with another variable that does have a relation with layout: the area of the development. Water depth is one of the main drivers of support structure design; too shallow or too deep waters are avoided, due to inaccessibility and cost or optimization constraints respectively. Water depths outside the range of 2 meters to 30 meters are not usually considered for an offshore wind farm (3), although it is technically feasible to design projects that place turbines outside this range. Seabed conditions are also quite important to be taken into account since they often determine which type of support structure is appropriate for the site.

Aghabi (3) synthesizes very clearly the influence of these variables on support structure cost: “If both the depth and the seabed terrain are homogeneous then the placing of the turbines in different places inside the perimeter will not influence the output cost. But if this is not the case, then the taking into account the foundation costs can be significant”. Most optimization routines and available software often assume homogeneous conditions, including Aghabi himself, who in order to handle the problem removes any unfeasible area from the solution space using an area restriction approach. Nevertheless, being able to optimize for non-homogeneous areas may be interesting for some developers.

3. The MZ Tool

3.1. Overview

The MZ tool was developed by Professor Michiel Zaayer of the Delft University of Technology, with its role being “similar to that of a wind turbine model in a wind tunnel for aerodynamic research. It is not the real thing, to be used in a commercial application. Rather, it is a prototype, with the purpose to generate knowledge” (2). Since the tool was conceived as a research prototype many common requirements for industrial use are not treated and thus it cannot be thought of as a direct competitor or alternative to the commercial packages presented in the previous chapter. The purpose of this chapter is to present the MZ tool, outlining its most important characteristics and features, which will then be cross-referenced with the user requirements to outline the target improvements of this project.

3.2. Inputs and Outputs

A very comprehensive way to assess the overall functionality and capabilities of a program is to analyze its inputs and outputs. A very simple model representing the function of a software (or of any tool for that matter) is shown in Figure 6.

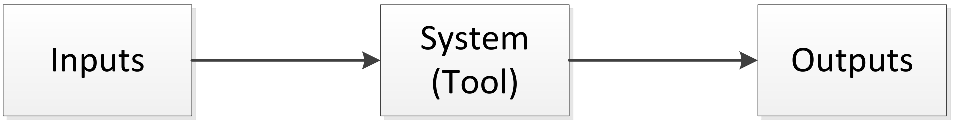


Figure 6. Simplified representation of a system in block-diagram form without feedback loops.

A tool can be visualized, in its most stripped-down conception, as a black box that converts inputs into outputs (which is also the strict definition of a system). It is often useful to first understand what is happening at both ends of the box before attempting to understand what happens inside it (which will be treated in the following section), since the user only has direct contact with inputs and outputs and thus will judge the performance on the tool based on them.

The MZ tool has a text-based interface that currently has no graphical input capabilities or options (due to being developed as a prototype). The user is allowed to modify three project inputs and their related parameters, which are described in Table 3.

Table 3. Allowed input description within the MZ tool.

Input	Description
Farms	Allows the user to define the target wind farm in terms of number of turbines per row and per column.
Sites	Serves as an input interface for all relevant site characteristics. It is composed of seven sub-modules, which are: wind climate, water levels, wave and current climate, water properties, geophysical properties, accessibility information and grid coupling point.
Turbines	Allows the user to define the turbine's characteristics in seven sub-modules, which are: geometric properties, mass properties, aerodynamic load properties, power properties, electrical properties, operational properties and financial data.

The "farms" input reveals that the tool is only capable currently of designing and optimizing wind farms with square layouts, defined by rows and columns with a certain number of turbines. This in turn also fixes the total number of turbines in the wind farm.

Site definition, while comprehensive in terms of the number of inputs that the user can customize, is strictly numerical. The wind climate is defined by the Weibull shape and scale parameters which are used to generate the corresponding distribution. Therefore, the user may not use personalized wind data. Wind rose or any other wind direction indicators are not supported either. Water depth is also assumed as constant throughout the whole site and is entered as a single numerical value, which is also the case for extreme tide and storm surge levels. The rest of the sub-modules also receive numerical inputs for all relevant parameters.

The possible wind farms that can be designed with the tool can only contain one type of turbine (although this is always the case for real projects, although in the future it may change), which is defined in the turbines input module. The most important parameters, such as rotor radius, mass of the RNA, drag coefficients (of idling rotor and nacelle), maximum thrust and total cost are all user-defined. The tool also supports the input of discretized thrust and power curves (the user must define the number of x-y points to approximate these curves).

Once all the inputs have been filled in the user is then given the option of generating and optimizing different wind farm designs using different combinations of sites, farms and turbines. The program then allows the user the selection between two operations: initial guess or optimization.

The initial guess option, as the name suggest, performs an initial calculation of all the parameters that characterize an offshore wind farm. The results are clearly not optimized; however, the option is useful to visualize expected project outputs and to help the user have a preliminary judgment of the expected results. The outputs of the initial guess are organized in six modules: system (overall wind farm performance), support structures, layout, electrical system, maintenance, and cost analysis.

While the inputs and outputs of the MZ tool provide great insight into its capabilities and limitations, some more specific information that is relevant to the present project can only be obtained by analyzing the inner structure of the tool.

3.3. Inner Structure

The MZ tool is composed of four main subcomponents, which are the application, the domain, the infrastructure and the user interface.

- The application is tasked with launching the tool.
- The domain contains most of the problem related variables, models, designers and analysts and thus constitutes the “heart” of the tool.
- The infrastructure, as was the case with the application, contains a single functional component intended to enable the tool to run: the file handler. Its name explains its function, which is to handle storage and retrieval of information generated in all other modules.
- The user interface contains all the build of the tool’s GUI.

For the purpose of this project most developments will be aimed at altering/generating the components included in the domain since all the wind energy related aspects of the problem are treated in it.

The MZ tool splits the overall wind farm design problem into disciplines, in a manner not different from that used in the present project. According to Zaayer (2) “after separation of the constraints, the remaining design problem of each discipline is to find the minimum of the LPC by variations of its own design variables, subject to constraints that also depend only on its own design variables”. The sequence of optimization per discipline as currently used by the MZ tool is presented in Figure 7.

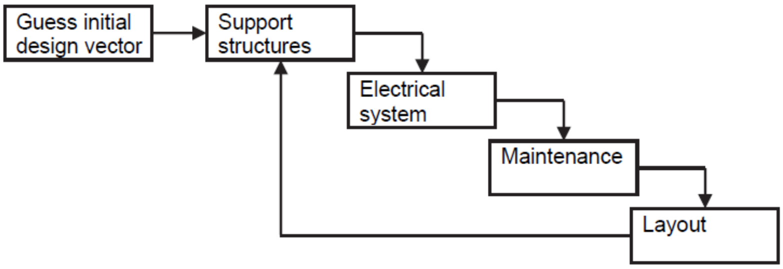


Figure 7. Sequence of disciplinary optimization, reproduced from (2).

A so-called multi-criteria analysis module sits at the center of the optimization routine (placed inside the humanities analysts), which is in charge of evaluating the objective function by obtaining all relevant parameters for the evaluation from the physics analysts. LPC is used by the MZ tool as the objective function, with its exact definition being contained in the economy analyst (also inside humanities).

The optimization routine itself is contained in the team designer, which makes calls to designers (classified by discipline); these in turn make calls to the individual sub-problem analysts. The routine is based on incremental improvement comparison of the objective function’s score, which is defined as:

$$score = LPC * LPC_{weight}$$

Equation 9.

The weight term is set as 1.0 as default; if a multi-objective optimization approach were to be implemented then different weights for different objective functions could be assigned via the same scoring system. The routine varies the parameters that affect the LPC, after which the objective function is evaluated and the score for that particular iteration is calculated. If the score of the previous run is saved, the increment (or decrease) in the score can be defined as a percentage of improvement. The variation of parameters continues until one of two conditions is met:

- The number of iterations reaches an allowed maximum, which is currently defined as 10.
- The incremental improvement between two separated runs is below 1% (stabilization).

This pattern search method may be classified as a local search algorithm (also called a univariate search) in which the problem is converted into a series of optimization routines.

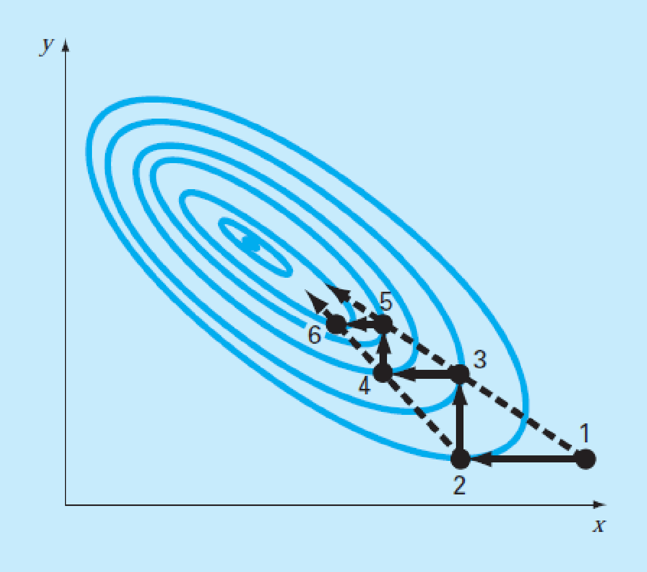


Figure 8. Schematic representation of univariate search. Reproduced from (4).

As a closing remark, Zaayer also identified three properties that have a high importance for the optimization tool and that should be retained as the prototype is refined and its functionality increases. In the first place, the tool should be able to produce solutions quickly, even at the expense of some accuracy in the solution. Secondly, the optimization routine should always be able to find a solution without any unnecessary intervention from the user (stable performance of the optimization). Finally, the determined LPC should not be highly sensitive to input parameters, in order to avoid instability in other optimization routines used by the MZ tool (not discussed here in detail).

3.4. Annotations by the tool's author

Zaayer (2) makes the following observations worth noting about possible functionality and performance improvements identified by reviewers of his tool (only the most relevant ones to the present project are presented):

- More options for the user, for example, having more freedom of choice regarding the selection of parameters which are currently preset (such as interest rate and cost parameters). Design variables of the wind farm (such as turbine siting) may also open to modifications by the user, although a problem of required expertise for changing such parameters arises.
- Automation of routine actions that the user will constantly perform, such as parameter variations and post-processing of different test scenarios.
- Increasing the scope of the application, for instance, to add functionalities that enable the tool to support detailed design or conceptualization and configuration phases.

- Include the assessment of fatigue of the support structure by extending the current fatigue model to include limit state analysis.
- Combat erratic response of spacing optimization.
- Currently the tool performs layout and support structure optimization separately, each bounded by its own constraints. If a fatigue assessment model is implemented the effect of wake turbulence would need to be considered to determine loads in the support structure, therefore, the division of optimization routines would no longer be acceptable due to the created bridge between both problems.
- Extending the tool to analyze different concepts and configurations of the project, in order to optimize a wind farm for a wider range of conditions.
- Addition of area restrictions and site-specific aspects.
- Visualization of inputs and outputs, as well as an overall improvement on the user interface allowing for file management and importing/exporting data from other programs.

All these remarks included in Zaayer’s own work will also be used in tandem with the identified functional requirements to identify the target problem.

4. User requirements and problem narrowing

4.1. Introduction to the Problem

Whenever a computational tool is to be developed determining the user requirements is a central part of the process, since they constitute the desired output that the software designer must somehow deliver based on problem inputs. In computer science the user's demands are often referred to as functional requirements.

In the layout design problem the user can be clearly identified: it is the wind farm planner, those actors who design the projects and then invest the capital to build them in order to gain future profit. However, getting a specific list of functional requirements is not very straightforward since contacting the users is not easy when being an outsider to the industry. Aside from that, planners often don't quite reflect on extended functionalities that computational tools may offer to them, thus making it even harder to determine exactly "what additional features they would like to have". For these reasons functional requirements must often be extrapolated, or reverse engineered, from actual projects, designs and general outputs from the client itself. Analyzing what current tools offer is also a useful way to paint a general picture of what wind farm planners often expect from their computational tools.

There are three types of functional requirements, which are:

- Those that have already been addressed up to a satisfying degree.
- Those that have been addressed but still leave some room for improvement to completely fulfill the user's demands.
- Those that remain completely untreated, due to either lack of knowledge on how to or due to the user's failure to identify "hidden potential" in additional requirements.

This chapter pretends to use some of these extrapolation methods to try and define what the functional requirements of wind farm design software are. These will then be prioritized, based not only on the importance of having said functionality but also on how achievable the implementation would be based on current knowledge and the scope of this project.

The methodology presented to obtain the final list of functional requirements was as follows: first, different approaches to extrapolating said requirements from available sources, tools and general observation were used (sections 4.3 to 4.6). All notable observations, features and other relevant information was then translated into requirement form which were then grouped, categorized and reshaped to avoid any redundancies (section 4.7). Then, this final list of requirements was evaluated according to certain indicators and figures of merit to obtain the final prioritized list. This process if further described in section 4.8.

4.2. Commercially available wind farm design tools

There are a number of software tools available in the market automatic layout design and optimization capabilities. The functions of each one of them shall be briefly described in this section, in order to further elaborate on the state of the art of wind farm design tools and help identify aspects in which the MZ tool may be lagging behind with respect to said software solutions. It is worthwhile to note that most of these tools have been developed for onshore wind farm design, which means that some of their functions may not be relevant for offshore wind power. This section briefly presents them, although a more detailed analysis of their capabilities and characteristics is given in sections 4.2.5 and 4.3.

4.2.1. GL Garrad Hassan WindFarmer

This program, developed by one of the world's biggest renewable energy consulters, is one of the most complete software packages for wind farm design and optimization. Its functions are divided in modules, which include energy, financials and electrical calculations as well as visualization and rendering capabilities, among others. More information on WindFarmer can be found in (50).

4.2.2. AWS Truepower openWind

AWS Truepower is a renewable energy consultancy, with around three decades of experience. Their wind farm design tool, named openWind, comes in two versions: a deluxe Enterprise edition and a Basic one, with more limited functionality. The tool uses a GIS (Geographical Information System) approach to the layout problem, meaning that the use of maps, digital terrain models and layering techniques are the core of the program. More information on openWind is available in (38).

4.2.3. EMD WindPro

Developed by EMD International A/S, which is a software and consultancy company in green energy projects based in Denmark, WindPro is probably the most robust wind farm design and optimization tool available in the market (although it can be argued that WindFarmer may come in a very close second place, if not being an actual tie). The software provides a very wide array of modules and tools that take into consideration virtually all aspects that are elated to wind farm development. Analogous to WindFarmer, WindPro is also organized by modules whose hierarchy and main functions are depicted in Figure 9.

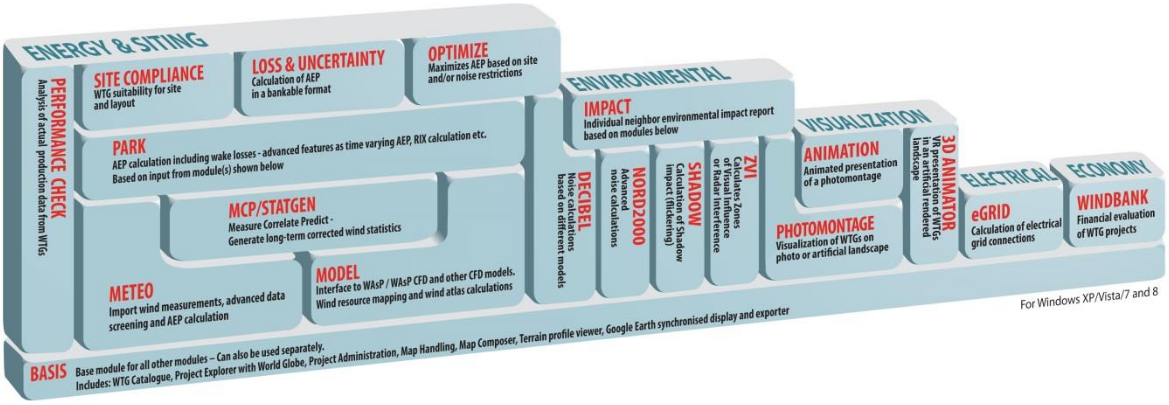

ENERGY & SITING
PERFORMANCE CHECK
Analysis of actual production data from WTGs
SITE COMPLIANCE
WTG suitability for site and layout
LOSS & UNCERTAINTY
Calculation of AEP in a bankable format
OPTIMIZE
Maximizes AEP based on site and/or noise restrictions
PARK
AEP calculation including wake losses - advanced features as time varying AEP, RIX calculation etc.
Based on input from module(s) shown below
MCP/STATGEN
Measure Correlate Predict - Generate long-term corrected wind statistics
METEO
Import wind measurements, advanced data screening and AEP calculation
MODEL
Interface to WAsP / WAsP CFD and other CFD models.
Wind resource mapping and wind atlas calculations
ENVIRONMENTAL
IMPACT
Individual neighbor environmental impact report based on modules below
DECIBEL
Noise calculations based on different models
NORD2000
Advanced noise calculations
SHADOW
Calculation of Shadow impact (flickering)
ZVI
Calculates Zones of Visual Influence or Radar Interference
VISUALIZATION
ANIMATION
Animated presentation of a photomontage
PHOTOMONTAGE
Visualization of WTGs on photo or artificial landscape
3D ANIMATOR
VR presentation of WTGs in an artificial rendered landscape
ELECTRICAL
eGRID
Calculation of electrical grid connections
ECONOMY
WINDBANK
Financial evaluation of WTG projects
For Windows XP/Vista/7 and 8
BASIS
Base module for all other modules – Can also be used separately.
Includes: WTG Catalogue, Project Explorer with World Globe, Project Administration, Map Handling, Map Composer, Terrain profile viewer, Google Earth synchronised display and exporter

Figure 9. Short module description and classification within EMD WindPro. Reproduced from (51).

More information on WindPro is available in (51).

4.2.4. ReSoft WindFarm

ReSoft is a small UK-based company whose single product is the wind farm design software WindFarm. The tool is also map-based. Most of its features are shared by the programs previously presented. More information on WindFarm is available in (52).

4.2.5. Comparison between available software

Using the main characteristics that each of the four most widely used wind farm design tools offer an objective comparison was elaborated. Note that this is only based on information provided by the manufacturers on their official websites. Table 4 and Table 5 present a feature detailed comparison of all commercially available software tools.

Table 4. Feature comparison (optimization and environmental variables) for commercially available wind farm design software.

	Objective Functions			Environmental				(-)
Program	COE	Yield	Noise	Flicker	Noise	Wildlife	Visual	Non-random layout optimization
GL Windfarmer	x	x	x	x			x	?
AWS	x		x	x	x		x	x

openWind								
EMD WindPro		x	x	x	x		x	x
ReSoft WindFarm	x	x		x	x		x	x

Table 5. Feature comparison (other concepts) for commercially available wind farm design software.

	Other Concepts						
Program	Wind data management	Uncertainty analysis	Loading checks	Economic analysis	Electrical checks	GIS/mapping support	Turbulence Intensity
GL Windfarmer	x		x	x	x	x	x
AWS openWind		x				x	x
EMD WindPro	x	x	x	x	x	x	x
ReSoft WindFarm	x					x	

The most important conclusions that can be drawn from this feature comparison, as well as from general characteristics, are:

- All analyzed tools give the designer the choice of at least two objective functions for the layout optimization process. The choice in commercial software is limited to the three functions presented in Table 4.
- None of these commercial tools has been specifically developed for offshore projects. This is especially notable in the emphasis the available software tools place on variables such as noise level evaluation and shadow flicker, which have no relevance for offshore wind projects.
- None of the tools assess environmental impacts on wildlife, which can be relevant for both on and offshore projects (although the evaluation methods for both cases and the extent of the effects are remarkably different).
- Layout optimization routines play a central role in all four tools analyzed in the review. All of them are able to perform optimization while adhering to area, boundary or

environmental constraints; an exception is GL WindFarmer although it may be because Garrad Hassan may not have published this information regarding their software, thus it is highly likely that this capabilities are included as well.

- Visual impact assessment has also become an industry standard, and some of the methods used for onshore projects such as ZVI, photomontages and rendering may translate correctly to the offshore case (although these are only good for a subjective valuation).
- Most of the currently used tools are GIS based, which gives them the advantage of being able to work with real multi-layered maps and use a database with very particular and often relevant site information. Results visualization is also simplified by using this kind of approach.
- Uncertainty analysis is an interesting tool to implement for giving the designer a more realistic appraisal of the project since risk management often plays an important role in project funding.
- While economic viability analysis is often considered to be a centerpiece of large scale project planning two of the evaluated tools don't have a built-in module for said purpose. Although these two programs (openWind and WindFarm) are both able to use cost of energy as an objective function, which is an inherently economic variable, they are not able to perform more advanced financial calculations such as NPV, IRR, liquidity, expected revenues, amongst other possibilities.

4.3. Software Outputs

The first approach used to obtain functional requirements follows a very simple logic: if wind farm planners are buying and using commercially available software it is because the solutions they offer comply with their demands (at least to a certain point). Therefore, analyzing said existing programs in more detail is a very straightforward way to reverse engineer the expectations their users have.

In order to do this extrapolation the main features of four commercially available programs previously considered (section 4.2) will be used as the analysis base. The objective now is to translate each program's features to "what is the user looking for" or "why I, as a wind farm planner, would find this useful". The approach is presented in user-thoughts diagrams, which are presented below. In these diagrams all the elements that start with "…" should be read as "I would like…", since this is the perspective of a user that acquires and uses the software.

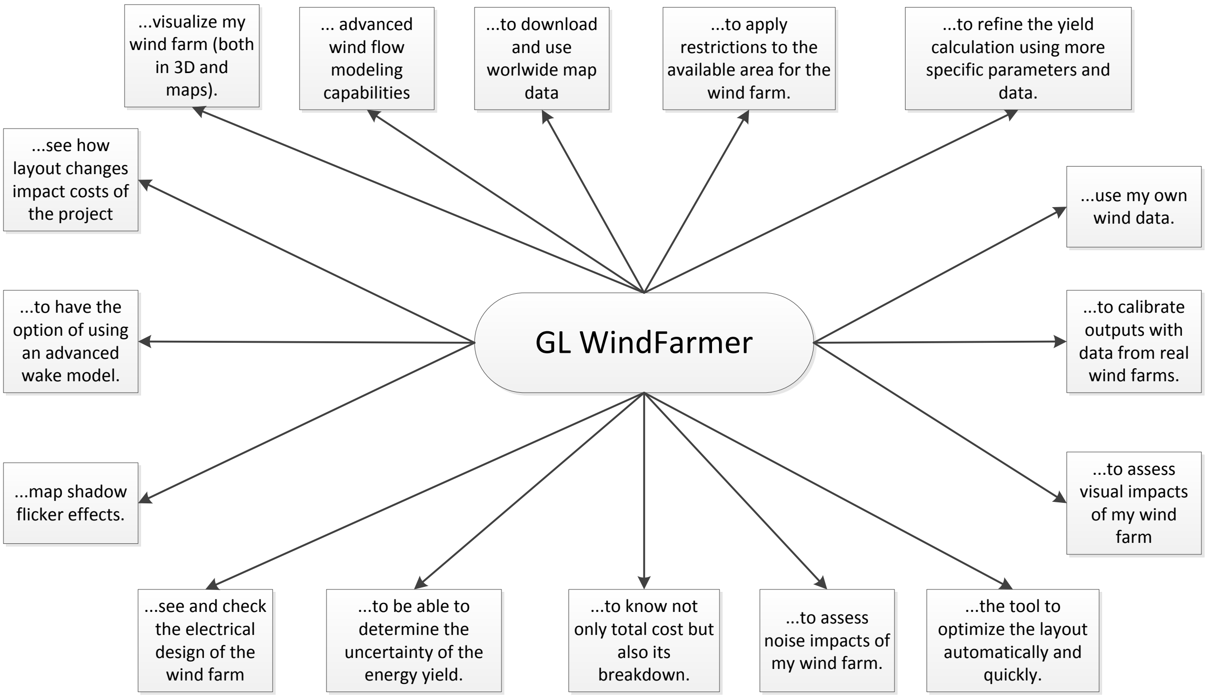

GL WindFarmer
...visualize my wind farm (both in 3D and maps).
... advanced wind flow modeling capabilities
...to download and use worlwide map data
...to apply restrictions to the available area for the wind farm.
...to refine the yield calculation using more specific parameters and data.
...see how layout changes impact costs of the project
...use my own wind data.
...to have the option of using an advanced wake model.
...to calibrate outputs with data from real wind farms.
...map shadow flicker effects.
...to assess visual impacts of my wind farm
...see and check the electrical design of the wind farm
...to be able to determine the uncertainty of the energy yield.
...to know not only total cost but also its breakdown.
...to assess noise impacts of my wind farm.
...the tool to optimize the layout automatically and quickly.

Figure 10. User-thoughts diagram for GL WindFarmer software.

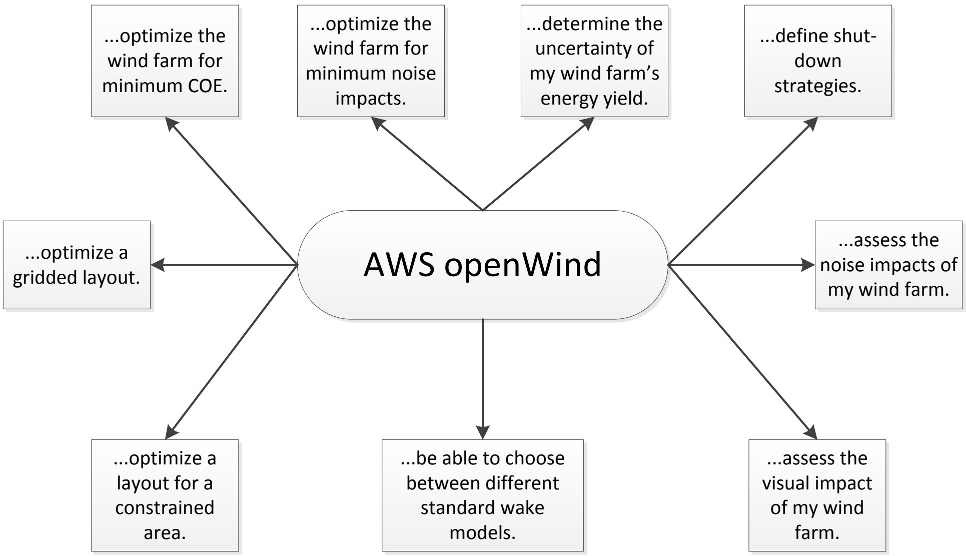
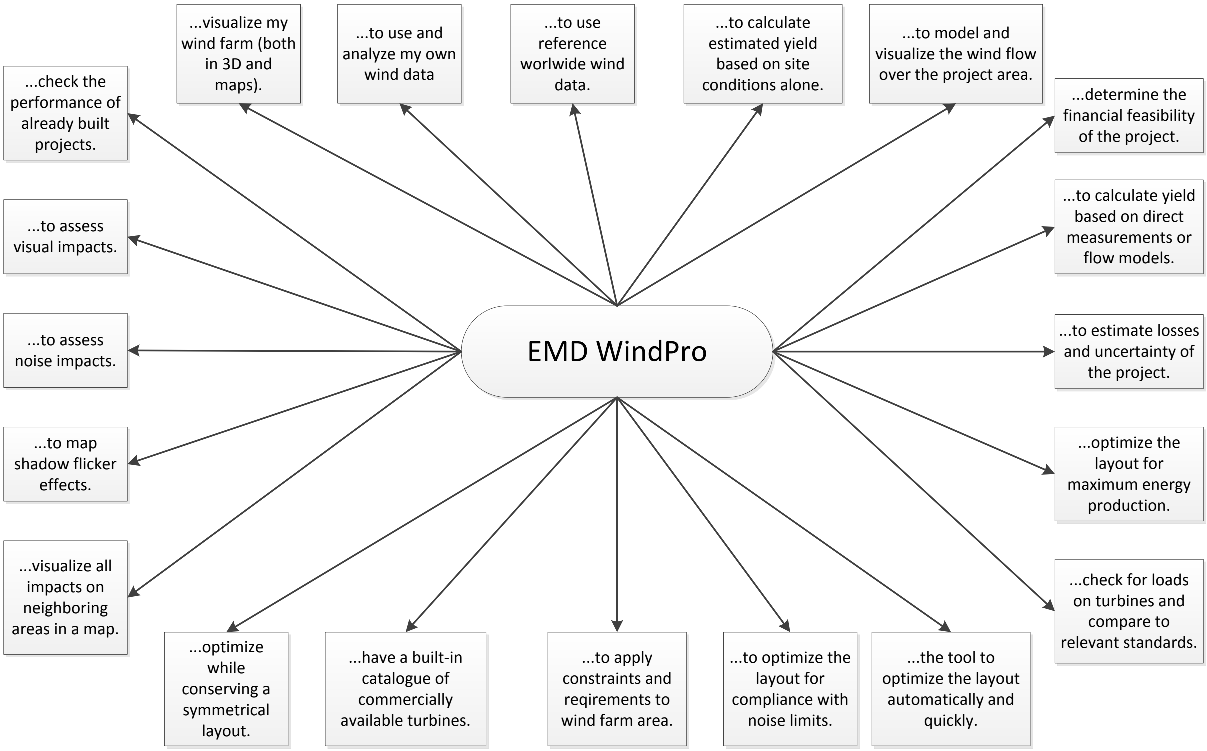
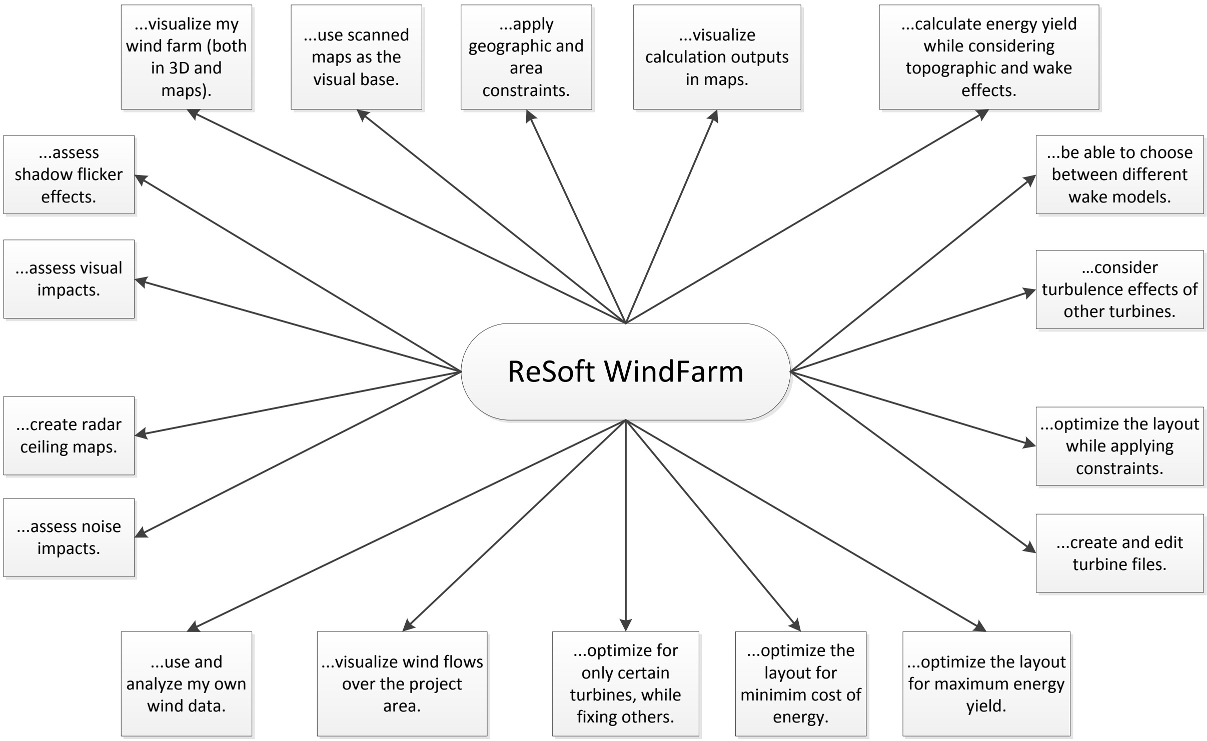

AWS openWind
...optimize the wind farm for minimum COE.
...optimize the wind farm for minimum noise impacts.
...determine the uncertainty of my wind farm's energy yield.
...define shut-down strategies.
...optimize a gridded layout.
...assess the noise impacts of my wind farm.
...optimize a layout for a constrained area.
...be able to choose between different standard wake models.
...assess the visual impact of my wind farm.

Figure 11. User-thoughts diagram for AWS openWind software.



...visualize my wind farm (both in 3D and maps).
...to use and analyze my own wind data
...to use reference worlwide wind data.
...to calculate estimated yield based on site conditions alone.
...to model and visualize the wind flow over the project area.
...check the performance of already built projects.
...determine the financial feasibility of the project.
...to assess visual impacts.
...to calculate yield based on direct measurements or flow models.
...to assess noise impacts.
EMD WindPro
...to estimate losses and uncertainty of the project.
...to map shadow flicker effects.
...optimize the layout for maximum energy production.
...visualize all impacts on neighboring areas in a map.
...check for loads on turbines and compare to relevant standards.
...optimize while conserving a symmetrical layout.
...have a built-in catalogue of commercially available turbines.
...to apply constraints and reqirements to wind farm area.
...to optimize the layout for compliance with noise limits.
...the tool to optimize the layout automatically and quickly.

Figure 12. User-thoughts diagram for EMD WindPro software.



...visualize my wind farm (both in 3D and maps).
...use scanned maps as the visual base.
...apply geographic and area constraints.
...visualize calculation outputs in maps.
...calculate energy yield while considering topographic and wake effects.
...assess shadow flicker effects.
...be able to choose between different wake models.
...assess visual impacts.
...consider turbulence effects of other turbines.
ReSoft WindFarm
...create radar ceiling maps.
...optimize the layout while applying constraints.
...assess noise impacts.
...create and edit turbine files.
...use and analyze my own wind data.
...visualize wind flows over the project area.
...optimize for only certain turbines, while fixing others.
...optimize the layout for minimim cost of energy.
...optimize the layout for maximum energy yield.

Figure 13. User-thoughts diagram for ReSoft WindFarm software.

User-thoughts diagrams are useful to think from the viewpoint of both the software publisher and the wind farm designer himself. Simple inspection of the diagrams also reveals which of the developer’s requirements are addressed by all of them (making them “standards” or minimum requirements), while also pointing out which features and aspects are considered only by some tools. If this is the case, then this second category of requirements are either (i) untreated or overlooked for the most part by other software designers, or (ii) not heavily demanded by the users and therefore not widely implemented.

The final step (yet a crucial one) in this analysis is the translation of the “user-thoughts” into actual functional requirements. For this particular method of extrapolation this task is quite simple since tool analysis reveals features and functions at a very detailed level of the problem itself and thus they can be taken directly as features that the user would like to have in his program. The final list of functional requirements (section 4.7) contains many of the features and needs identified using the user-thoughts diagrams, which appear unchanged from their initial formulation in this section due to their characteristic simplicity.

Another approach that is also useful is to find and use customer reviews of the software itself, which most of the time can be easily found online. However, due to the specialized function of the tools and the narrow user-group that they are aimed at no such reviews were available and thus this important feedback source could not be used.

4.4. Scientific literature

Literature can provide some insight in what the functional requirements may be. Research and development efforts are often focused on building knowledge and techniques that address problems whose solution is being demanded, therefore giving insight on the first two types of functional constraints listed in section 4.1. Note that referencing each particular requirement to a single author or source is an extremely complex task, since many of the identified requirements are not directly voiced by said sources or appear as part of the project background in most of the studied reports.

Additionally, the “future work” section often found in scientific publications can provide insight in the third type of requirements. The state-of-the-art report presented in section 2.2 gives an idea on where current research in layout optimization is heading at, from which some generalized user requirements can be extrapolated, as shown in Table 6 (note that these lists contains not only requirements for a project developer but also reflects interests of other actors).

Table 6. Extrapolated requirements from state-of-the-art study.

Category	Ongoing R&D	Reflected requirement
Optimization	Which optimization strategy is better suited to solve the layout	Obtain the highest-quality solution possible from a computational tool.

	problem?	
	Reduce computational time and burden by perfecting algorithms.	Efficient programs that solve the problem within reasonable time constraints.
	Which objective function is better suited for the optimization?	Develop and implement new objective functions which encompass all relevant problem variables and prove to yield good results.
Modeling	Developing and implementing better wake models.	Same as ongoing R&D.
	Developing and implementing better cost models.	Same as ongoing R&D.
	Use of economical valuation models and strategies.	Be able to know profits and benefits of investing in projects.
	Developing and implementing better cable models.	Improve loss estimation, and thus improve energy yield estimation.
	Improving cable topologies.	Reduce costs; improve transmission efficiency and overall yield.
	Implement automated support structure design.	Reduce costs, increase computational tool functionality.
Layout	Symmetry of layouts.	Increase public acceptance and thus probability of project success.
	Convert constraints into restricted areas.	Obtain more realistic solutions using a simple approach for the designer.
	Visual impact assessment and minimization.	Increase public acceptance and thus probability of project success.
Environmental	Assess climatic effects (local and global) produced by offshore wind farms.	Evaluate long term risks and consequences of mass deployment.
	Assess noise impacts and propagation.	Minimize burden on neighbors, thus increasing public acceptance.
	Assess impacts on local wildlife.	Evaluate long term risks and consequences of mass deployment.

Notice that as was mentioned before the table does not contain all the requirements that a project developer expects from wind farm developer software, it is simply a guide that in a further section will be combined with the other extrapolation methods to voice the final list of actual functional requirements.

The studied literature also provides some information regarding future work and opportunities that may be used to shape the expectations of those working on the field that still remain to be fulfilled. Since authors tend to voice and list their perceptions on what future work should be, clearly referencing each requirement to a particular source becomes a simpler task than with ongoing R&D activities and projects. The sources used for this analysis are now briefly presented, and the recommendations given by each were then grouped and translated to functional requirements.

Aghabi (3) and Pérez et.al (6) have developed works related to computational approaches to solve the layout optimization problem. Interestingly, both works contain the same choice of words to voice the authors' view on requirements and possibilities of improvement (it is unclear who wrote it first since both fail to quote the text): "There is still much work to be done in the field of wind farm optimization. These include wake modeling to properly predict the wind decay (and therefore the power losses). Other physical aspects such as foundations or cabling, human aspects as visual effects and area restrictions, and mathematical topics regarding the optimization approaches must also still be reviewed in detail". All of these requirements were also identified in the literature review and are therefore contained in Table 6. Aghabi's work, which consisted in the implementation of a simulated annealing optimization algorithm, also contains additional recommendations that may correspond closely to the developer's functional requirements.

Elkinton et.al (43) also mention possible additional functionalities and requirements for wind farm design software as part of their report concerning the OWFLO project. At the top of their priorities they voice the "need for software that can optimize an offshore wind farm layout based on a measure of the COE, such as the LPC. This type of software will be particularly useful as developers look at sites farther from shore and in deeper waters" (43). The OWFLO project was particularly concerned in developing a software tool that addressed specifically this need. Their work also mentions the lack of certain cost models that were developed by the authors when lack of sufficient literature was encountered. Since the reference used is a preliminary results assessment for the OWFLO project in this stage the authors mentioned that they were improving cost and component models, which corresponds well to some of the requirements identified previously.

A similar work to Aghabi's project is that of Mittal (24) in which he developed and implemented a genetic algorithm for layout optimization, named WFOG (Wind Farm Optimization using a Genetic Algorithm). In the list of recommendations included in his work Mittal identifies several improvements that can be extrapolated for any software development effort in the field.

Regarding the optimization routines and approaches perhaps no other work encompasses future work and still unsolved requirements better than the review by Khan et.al (6). Their work provides a very comprehensive and detailed list of research opportunities and remaining challenges specific for the iterative non-deterministic approaches that are the subject of their very comprehensive review.

All of the recommendations obtained from the sources mentioned in this subsection were translated to functional requirements, which are presented in Table 7. Note that only recommendations that lead to concrete functional requirements are considered.

Table 7. Functional requirements obtained from future work sections in literature.

Category	Author Remarks	Reflected requirement	Reference
Optimization	Faster optimization algorithm.	Achieve a solution in a reasonable amount of time.	(3)
	Possibility to use a heuristic approach.	Implement a heuristic optimization algorithm.	(3)
	Possible use of local search methods, such as Tabu search or GRASP.	Implement a local search optimization algorithm.	(3)
	Optimize based on COE.	Include COE or LPC as an objective function.	(43)
	Develop more efficient multi objective optimization approaches	Achieve a high quality solution in a reasonable amount of time.	(7)
	Increase focus on algorithm development and efficiency, including optimal parameter selection.	Achieve a high quality solution in a reasonable amount of time.	(7)
Modeling	Use of a more exact cost function.	Develop and implement better cost functions and models.	(3), (43)

	Evaluate the performance of different wake models to determine which is better.	Provide a choice of different wake models to the user.	(24)
	Analyze and implement the effect of turbulence on wake recovery.	Account for atmospheric turbulence effects.	(24)
	Study and implement a model that accounts for ground effect on wake expansion.	Develop better, more comprehensive wake models.	(24)
	Generate more realistic objective functions that consider only variables relevant to the layout.	Develop and implement better objective functions.	(7)
	Acknowledge and correct for the non-determistic nature of the algorithms.	Develop a strategy that minimizes randomization errors that stem from the heuristic nature of the algorithms.	(7)
Layout	Managed to perform layout optimization while adhering to a certain predefined pattern. He mentions that his approach, while functional, may still be greatly improved.	Ability to generate optimized symmetrical (or slightly optimized) turbine layouts.	(3)

4.5. Layouts and constraints

A very obvious way to extrapolate user requirements is to analyze what their final products look like; in the case of the present project this consists in understanding how final layouts of wind farms are in real life. For this analysis two already built offshore wind farms were considered: the Egmond aan Zee (OWEZ) off the coast of the North Holland province in the Netherlands and the Middelgrunden offshore wind farm, located 3.5 km away from the coast of Copenhagen, Denmark. The locations of the turbines were facilitated by a fellow student of the wind energy research group of the Delft University of Technology, Alberto Striedinger (53). In addition, a third planned offshore wind farm will also be considered, named the Q4 West project located in the vicinities of the OWEZ and Princess Amelia windparks in the

Dutch North Sea. The presented layouts (Figure 14, Figure 15 and Figure 16) have an aspect ratio as close to 1:1 as possible, which explains the odd formatting of the figures.

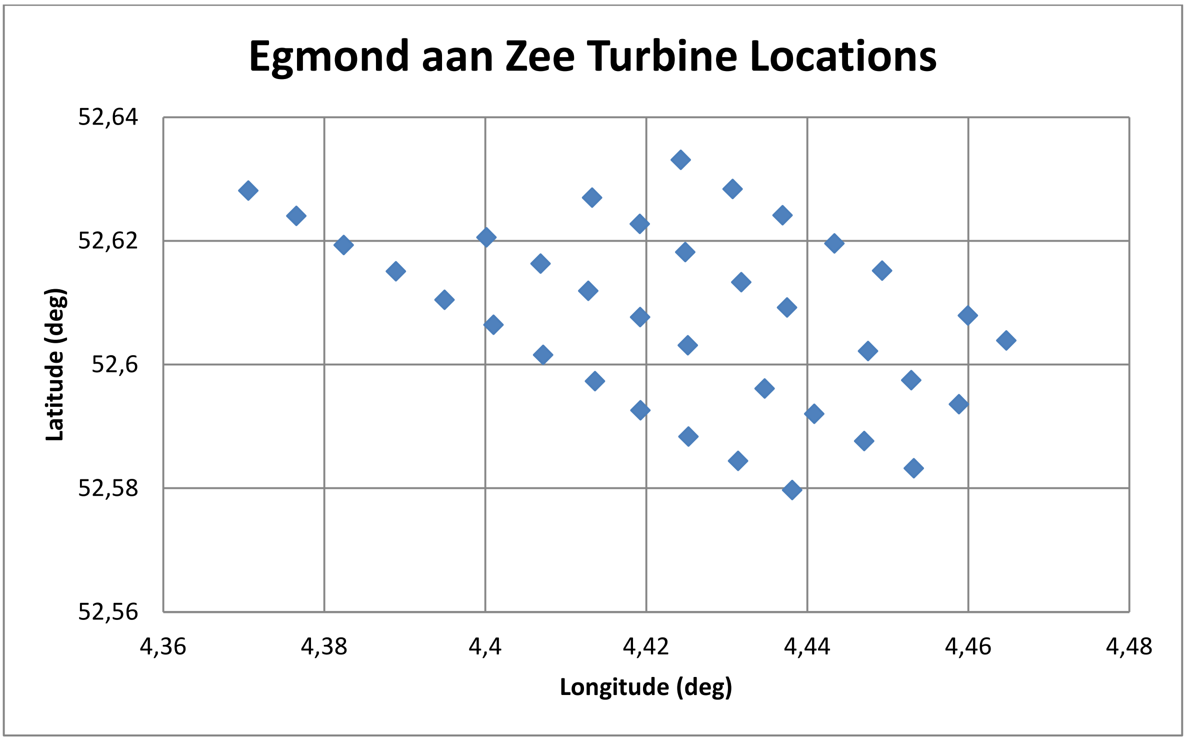


Figure 14. Turbine coordinates of the OWEZ wind farm. Reproduced from (53).

The turbine coordinates for the 36 generators that compose the layout shown in Figure 14 are not completely exact since they were obtained using a digitalization from an overhead view of the wind farm.

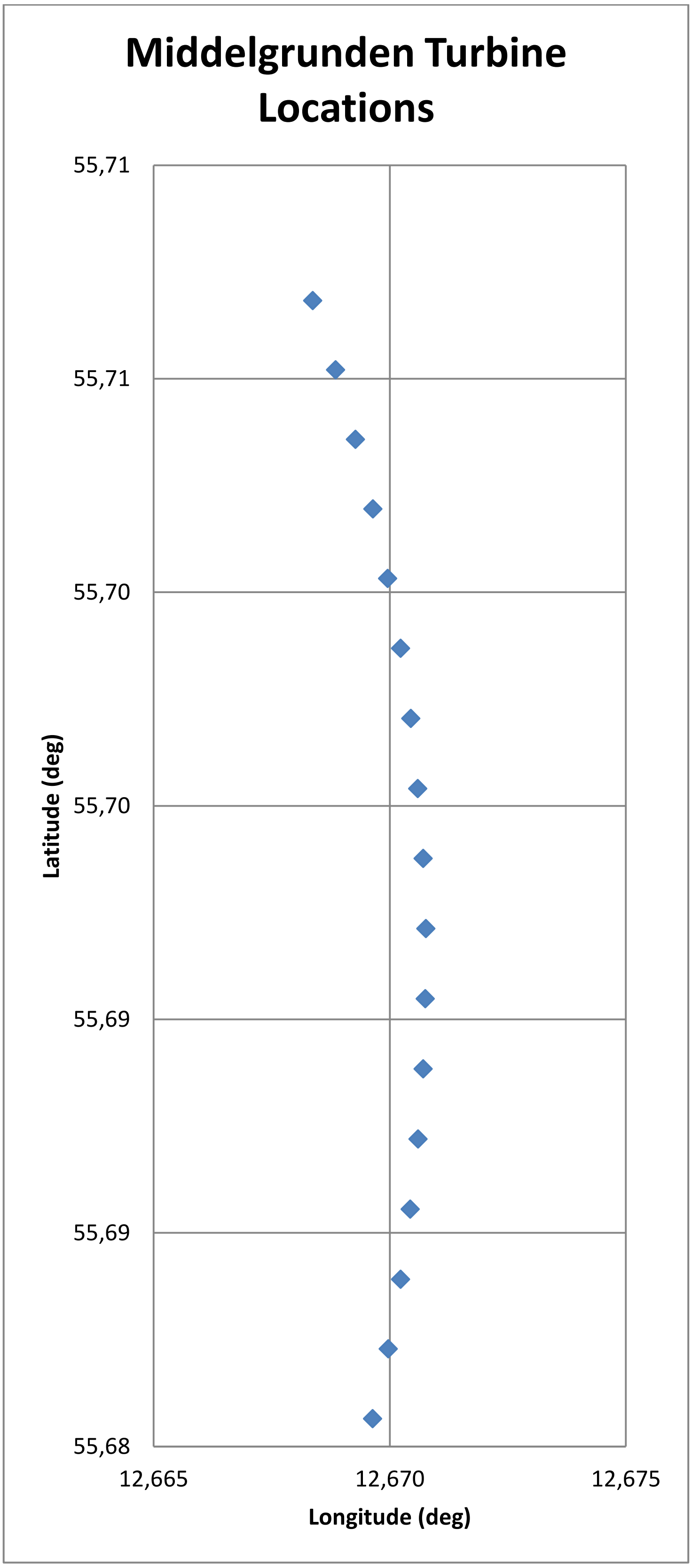


Figure 15. Turbine coordinates of the Middelgrunden offshore wind farm. Based on (53).

Figure 15 shows the turbine locations of the 20 generators that compose the Middelgrunden offshore wind farm. In addition to the two already built projects presented the Q4 West project will also be taken into consideration due to the following reasons:

- The layout, while organized, does not follow an exact geometric shape.
- The cabling layout is available.

The Q4 West project is being developed by Eneco. The proposed wind farm will be located 26 km away from the coast of Bergen aan Zee.

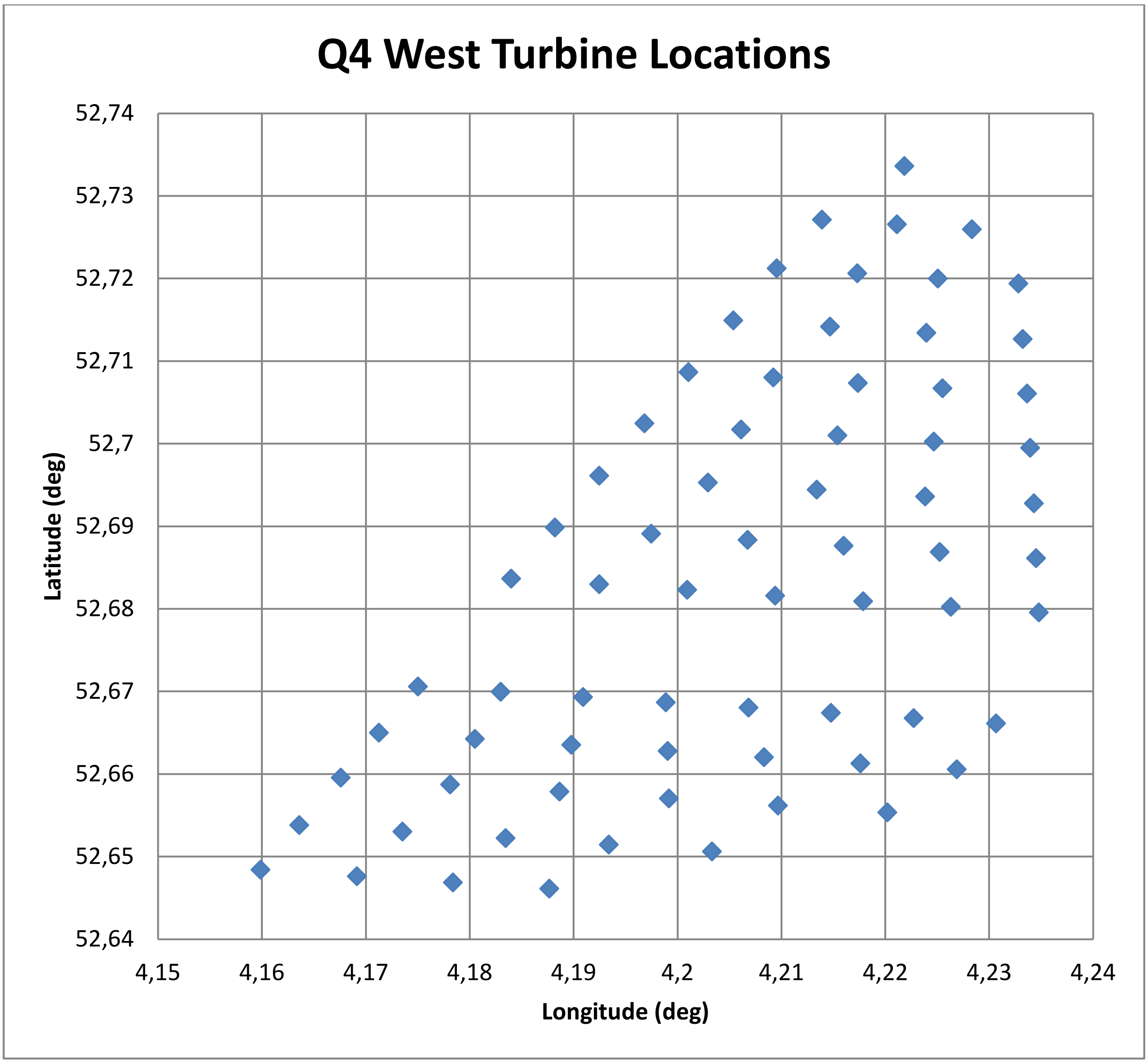


Figure 16. Turbine coordinates of the Q4 West planned offshore wind farm. Data from (54).

The report by Pondera consulting for Eneco (54) also contains the proposed electrical infrastructure for the wind farm as shown in Figure 17.

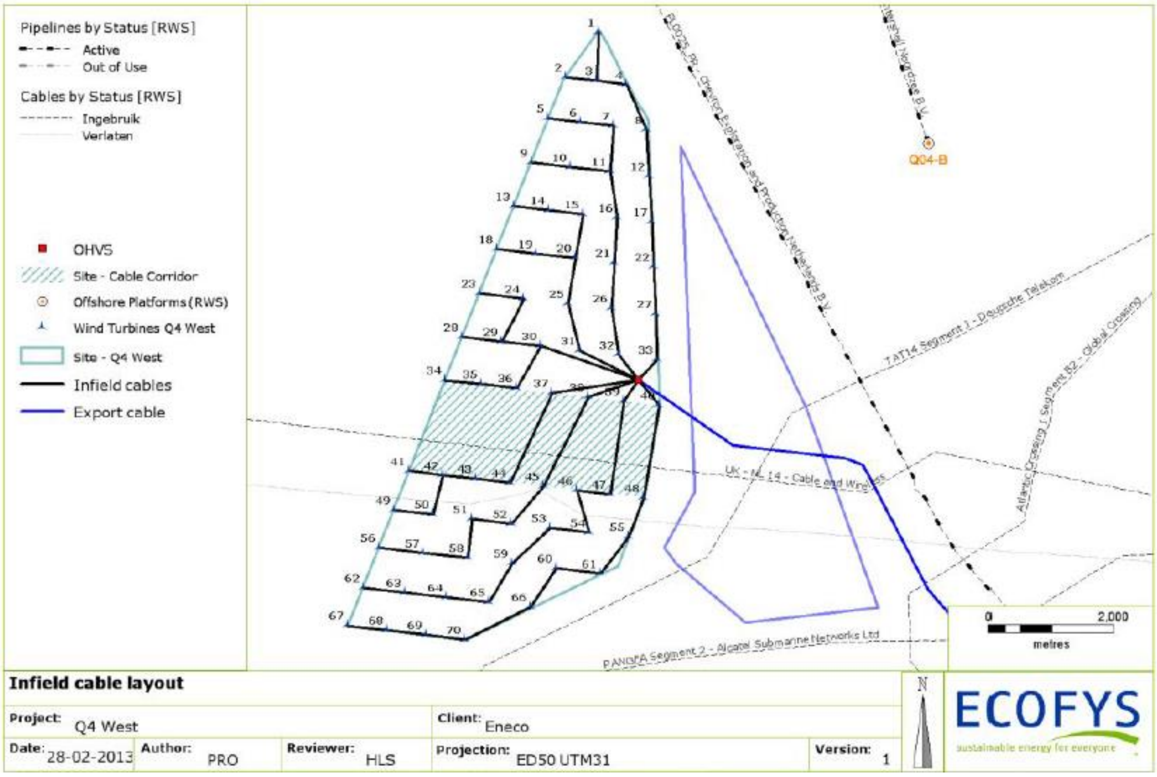


Figure 17. Proposed electrical infrastructure for the Q4 West offshore wind farm. Reproduced from (54).

Some important conclusions may be drawn from simple visual inspection of these layouts that reflect clearly what offshore wind farm planners aim for:

- The layouts of the OWEZ, Middelgrunden and Q4 West projects clearly demonstrate the developer's tendency to achieve a symmetrical or at least mostly organized layout. OWEZ demonstrates a more traditional rectangular array (save for an extended arm) while the Middlegrunden wind farm has an arching disposition of turbines. Even the planned Q4 West wind farm seems to follow a "boomerang" type of pattern, with lines of turbines clearly defined inside an irregular area.
- Area limitations and constraints can be clearly identified in all three analyzed layouts. For the OWEZ wind farm a gap between turbines can be observed clearly (starting from approximate coordinates [4.43, 52.6] and ending in [4.45, 52.61]). The reason for the existence of this gap is a trench in the ocean floor, which results in a section within the development area with a higher depth. This area was probably not considered as a viable place for turbine positioning and thus the layout was designed without taking it into account. The Middelgrunden wind farm is arguably the most aesthetically pleasing out of the three analyzed projects (arguably because said statement is subjective). This may be due to the fact that it is closer to a human settlement than both OWEZ and Q4, meaning that visual impact minimization was probably a higher priority during its design phase, reflecting thus the need to include

said aspect as a requirement. The Q4 West wind farm also displays an area constraint, evident in the no-siting band that cuts the layout from east to west. The presence of a submarine cable, clearly visible in Figure 17, demonstrated another type of area restriction that has a direct impact on the layout.

- The distance between turbines remains an important parameter to be respected, no matter what kind of layout is chosen. The coordinates may be used to check that the minimum recommended separation of 4D (6) is amply respected by all the layouts considered.
- The cabling layout presented in Figure 17 showcases important design drivers that planners often take into account. For instance, the Q4 West clearly has an electrical infrastructure of the star type (as shown in Figure 5-IV) with the turbines connected in branches and a central transforming station (OHVS, standing for Offshore High Voltage Station). There is a single export cable to transport the energy to the shore connection. Also noticeable is the fact that the topology follows logically the proposed layout by minimizing the cabling distance. For this particular wind farm up to 10 turbines are allowed per branch of cable, which is below the recommended maximum of 19 (55).

All the observations here obtained must be converted to functional requirements, analogous to the other methodologies presented in this chapter.

Table 8. Extrapolated requirements from actual layouts and constraints.

Observation	Reflected Requirement
Symmetrical (or somehow organized layouts)	Ability to generate optimized symmetrical (or slightly optimized) turbine layouts.
Area restrictions and obstacles have an impact on overall layout design.	Restrict available turbine siting area due to various constraints.
All analyzed wind farms respect a minimum turbine separation.	Apply a constraint of minimum turbine separation.
Minimizing cabling distance in design of cable topology.	Implement an automated cable topology optimizer, closely related to the layout optimization routine.

4.6. Extrapolation from objective functions

This method of requirement extrapolation consists in breaking down the objective functions used for the optimization routines into their most basic components. The logic behind it is quite simple. As the name suggests, objective functions are the main targets that routines strive to maximize or minimize, therefore, they contain most of the problem variables in their formulation. It is therefore possible to reverse engineer not only functional requirements but also existing conditions and inputs for the tool by breaking down the objective functions into their building blocks.

4.6.1. Energy yield breakdown

Amongst all the available objective functions for the layout optimization problem energy yield is probably the easiest one to decompose. This is due to the fact that is a purely technical function (no cost dependency), therefore cost models are not required to assess maximum yield.

Energy yield is, in its most fundamental definition, given by the following equation:

$$E_y = T \int_{U_{ci}}^{U_{co}} P_{el}(U) * f(V)dV$$

Equation 10.

where *T* is the number of hours per year, U_{ci} and U_{co} are the cut-in and cut-out wind speeds of the turbine, respectively, $P_{el}(V)$ is essentially the power curve and *f(V)* the frequency distribution of the wind speed. Further decomposition of the function is possible by splitting each term of the equation into more basic components. These breakdowns are presented in the form of breakdown diagrams. However, the decomposition here presented is not aimed at reaching the most elemental level of the structure of the function; for example, understanding that the power curve is needed to assess energy yield is a sufficiently detailed piece of information, thus, it is not necessary to break the power curve into its building blocks. The target level of detail here treated was determined by the author’s judgment.

The energy yield is the simplest of the objective functions analyzed, thus making it possible to present a total breakdown diagram for the function. When additional breaking down of an element is required it will be treated separately.

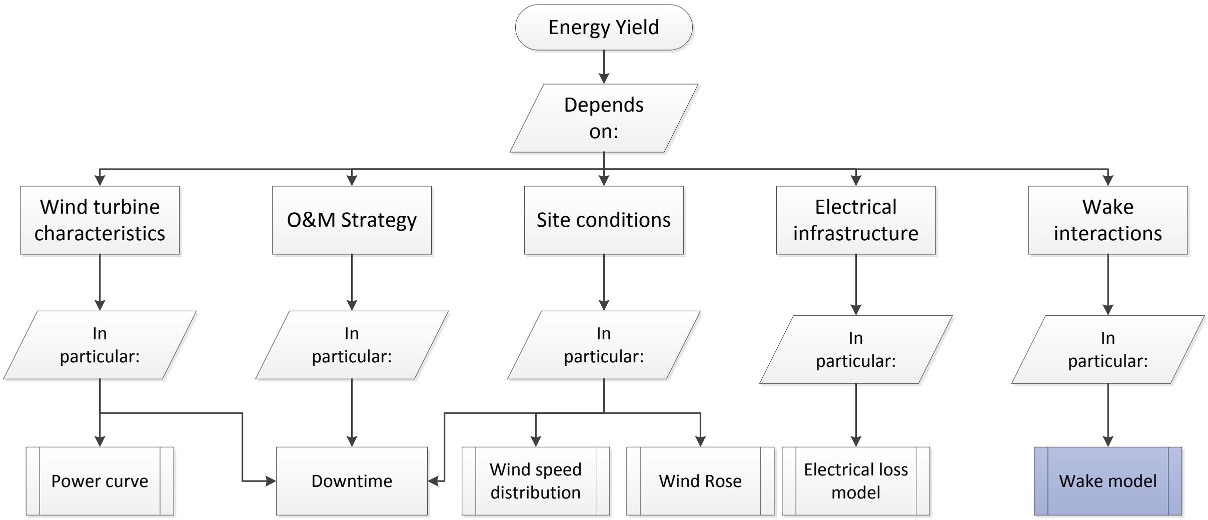


Figure 18. Energy yield function breakdown diagram.

Figure 18 shows the decomposition into basic components required for energy yield evaluation for a complete wind farm. The power curve encompasses all relevant information regarding the undisturbed energy production of the wind turbine, and it indirectly also contains information regarding its aerodynamics, rotor area, etc. By combining this power curve with the site conditions (in particular with the wind speed distribution) the undisturbed yield of a single turbine may be obtained, as described by Equation 10. The wind speed distribution also encompasses the effect of the hub height on the total yield, since it is measured or given at said height. The wind rose allows the evaluation of this output for all desirable wind directions.

Finally, the results for single turbines must be extended to wind farm scale. Wake models are the missing link in the problem, since they account for the interactions between turbines inside the wind farm in the form of wake losses. In order to achieve an accurate prediction of the energy yield, wake effects need to be considered.

What is left in the lower level of the function breakdown diagram are the required “gears to make the machine work”, that is, requirements that the program need to somehow have built-in in order to use energy yield as the objective function. These lower blocks are those represented by two vertical lines enclosing the inner text (notice that downtime is not one of them since it requires additional breaking down). Components that affect or are affected the most by the layout design have been colored blue to highlight the relevance to the current problem. For the energy yield, only the wake model is affected by the layout, since it must be able to account for wake interactions between different turbines. Also, wake interactions are necessary to calculate power losses, which in turn affect the total energy yield directly.

From the analysis of the energy yield function the identified functional requirements are presented in Table 9.

Table 9. Requirement identification from the energy yield objective function analysis.

Component	Reflected Requirement
Wake model	Include a robust wake model that can handle any layout.

4.6.2. Cost of energy breakdown

The cost of energy (COE), also known as levelized production cost (LPC) is a more complex function than the energy yield since it includes financial considerations into the optimization, therefore requiring the use of additional models and variables. COE is a more comprehensive function than energy yield, especially from a wind farm developer's point of view.

Equation 3 is the most common formulation for the LPC of a wind farm. Four variables are included in the formulation: investment costs (C_{inv}), operation and maintenance costs ($C_{O\&M}$), energy yield and a so-called annuity factor (*a*). The energy yield was explored deeply in section 4.6.1, thus its analysis is omitted from this section.

Investment Costs

Investment costs are usually expressed using cost models that are a function of the problem design variables. Investment costs for an offshore wind farm can be decomposed according to Equation 11.

$$C_{inv} = C_{ei} + C_t + C_{fou} + C_{ins} + C_{oth}$$

Equation 11.

where C_{ei} are the electrical infrastructure costs, C_t are turbine costs, C_{fou} are the costs of the foundations and support structures, C_{ins} are the installation costs, and C_{oth} are other costs not included in previous categories. These sub-costs, in turn, can be further decomposed resulting in the breakdown diagrams presented below.

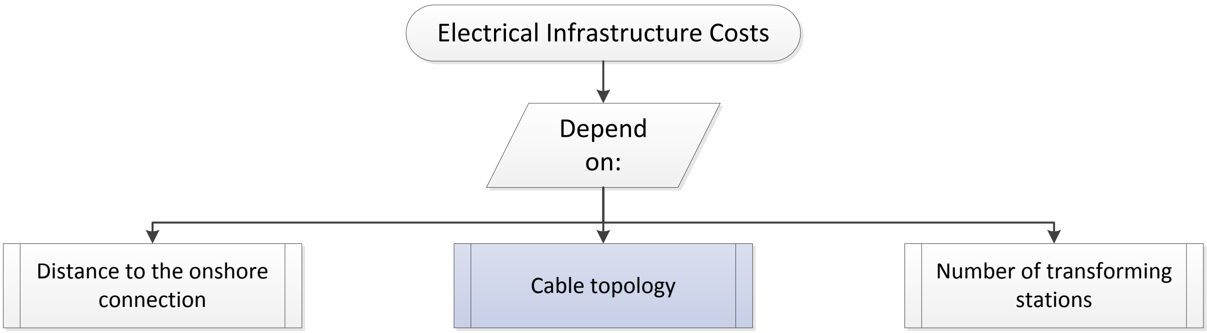

Electrical Infrastructure Costs
Depend on:
Distance to the onshore connection
Cable topology
Number of transforming stations

Figure 19. Electrical infrastructure costs breakdown diagram.

Amongst electrical infrastructure costs only the cable topology is affected by the layout, since cabling distance and overall design of it is a function of the turbine positions (the Q4 West layout and its cable topology evidence this fact).

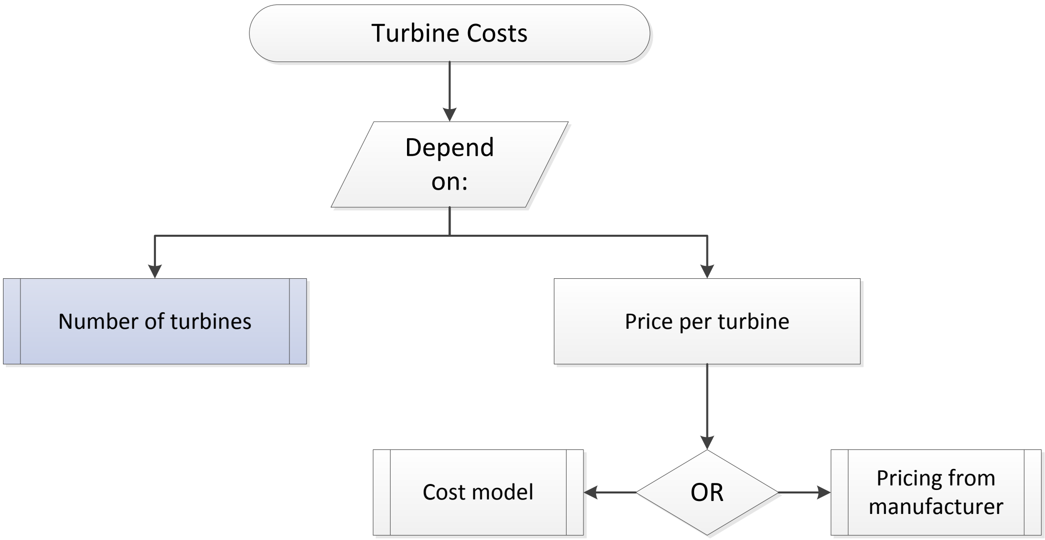


Figure 20. Total turbine costs breakdown diagram.

The breakdown reveals that the number of turbines is related to the layout, although the relation is not straightforward: the layout may be a function of the number of turbines, but the opposite can also hold (number as a function of layout). This depends entirely on how the variable is treated. If the number of turbines is a fixed input then the final layout will be a function of it; however, if the optimization has as a side goal determining the optimal number of turbines then the dependence becomes somewhat unclear. The discussion presented here is not aimed at determining which is the best way to treat the number of turbines variable, but rather to acknowledge its relevance to the layout optimization problem and therefore considering its role as part of a functional requirement.

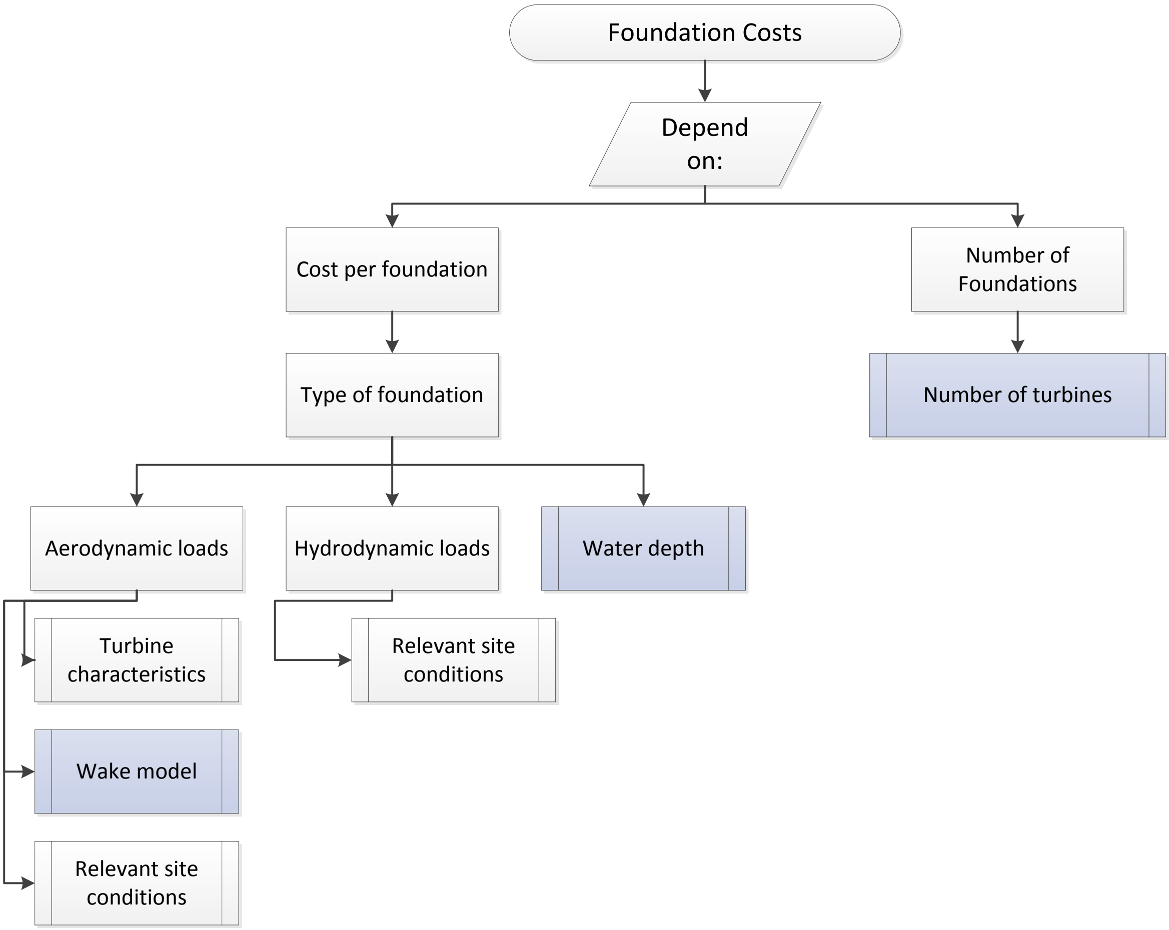


Figure 21. Foundation costs breakdown diagram.

Figure 21 reveals that foundation costs are significantly more complex than turbine and infrastructure costs, since the choice of a particular foundation type (monopile, jacket, tripod, etc.) is dependent on a large number of variables. The diagram identifies three components that are a function of the layout, two of which were already previously identified. Water depth appears as relevant since it is related to the area and turbine siting problem: if the layout places a turbine in deep water then the support structure costs will increase due to this selection. The area constraint problem is also closely tied to water depth considerations, since often designers opt to exclude too deep or too shallow zones from the available solution space.

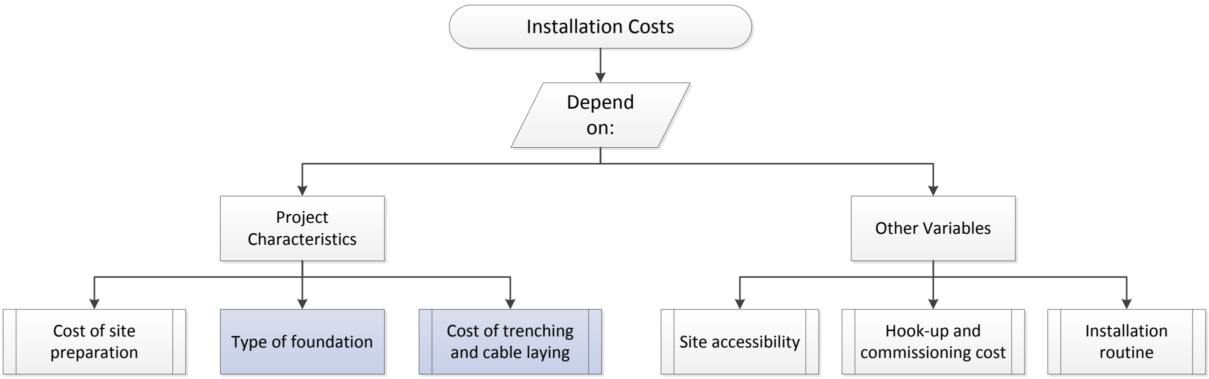


Figure 22. Installation costs breakdown diagram.

The installation costs breakdown reveals only two components that are layout-dependent. The type of foundation was decomposed (Figure 21) in the foundation costs breakdown, revealing its indirect dependence to layout related variables. The second component is the cost of trenching and laying the cables on the seabed, which is a function not only on the total cabling distance but also on the soil type and terrain characteristics (all related to the area selection problem).

Operation and Maintenance Costs

O&M costs are not as complex as installation costs. The breakdown diagram in Figure 23 presents the individual components that influence the total O&M costs.

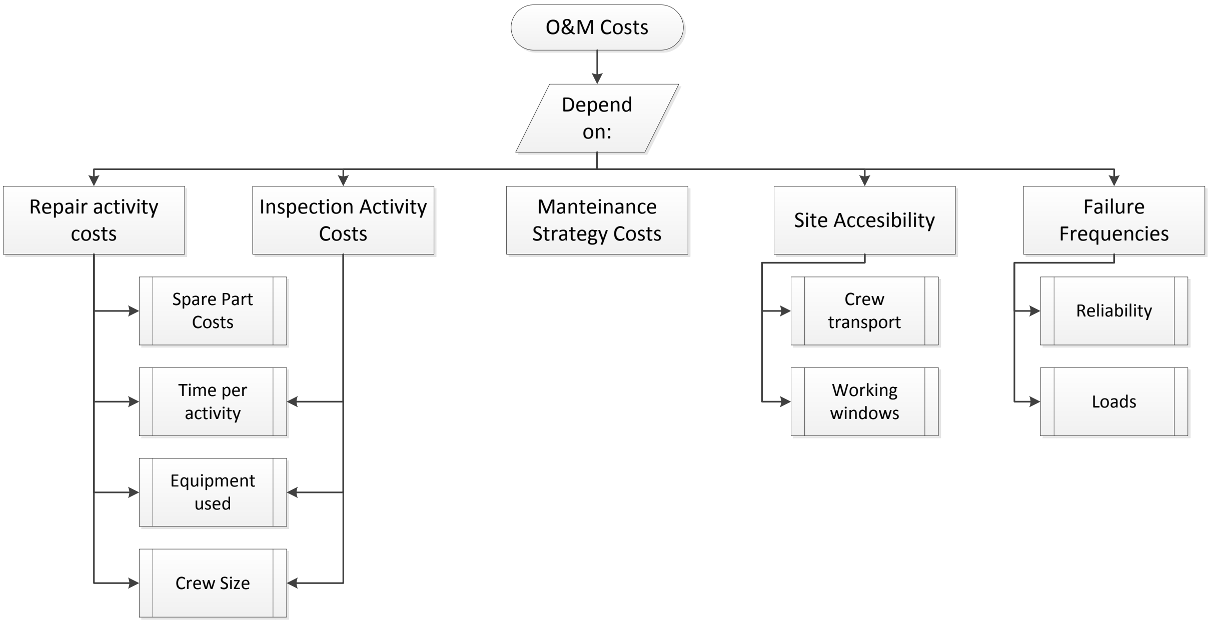


Figure 23. O&M costs breakdown.

The diagram shows that O&M costs are completely independent on the layout chosen; they are instead heavily influenced by the site conditions and the strategy implemented to perform the respective actions.

Annuity Factor

The final term appearing in the COE equation is a financial parameter often used in project valuation called the annuity factor, which is used to calculate the present value of future payments (aptly name annuities), all of which are equal over a certain time frame. The annuity factor is given by the following equation:

$$a = \frac{1 - (1 + r)^{-n}}{r}$$

Equation 12.

Where *n* is the number of terms or periods and *r* is the interest rate per period. It is evident that the annuity is exclusively a function of how the project is being financed, and it can often be calculated with high accuracy if a proper choice if interest rate is made.

Extrapolated Requirements

Using all the detailed breakdowns of the individual components of the COE objective function, additional functional requirements related to the layout may be obtained. These are reported in Table 10.

Table 10. Extrapolated functional requirements from the cost of energy objective function.

Component	Reflected Requirement
Wake model	Include a robust wake model that can handle any layout.
Cable topology	Implement an automated cable topology optimizer, closely tied to the layout optimization routine.
Number of turbines	Determine the optimal number of turbines to be installed, or optimize for a fixed number of turbines.
Water depth	Ability to optimize for a site with variable water depth.
Cost of trenching and cable laying	Develop and implement better cost models that offer a more detailed breakdown of every contributing factor.

4.6.3. Profit breakdown

The profit function proposed by Réthore et.al (40) as part of the TopFarm framework is, as described by Tesauro et.al (30) the most comprehensive objective function available. The equation that describes the financial balance presented in section 2.3 shows that the function is heavily influenced by financial parameters, to an even greater extent than the COE/LPC function. A detailed analysis of Equation 4 allows for the decomposition of the profit function.

Figure 24 presents the function breakdown diagram for the financial balance, or profit function. It shares many common requirements and building blocks with the COE function; the operation and maintenance costs are broken down in the same way for both objective functions. Noticeable is the lack of installation costs in the evaluation of the profit function, which the COE does consider. However, many of the considerations made to assess the investment costs are also taken into account in the profit function, since an important part of the evaluation is the determination of the future value of the investments made at the start of the project, which means that all relevant cost models must be also known.

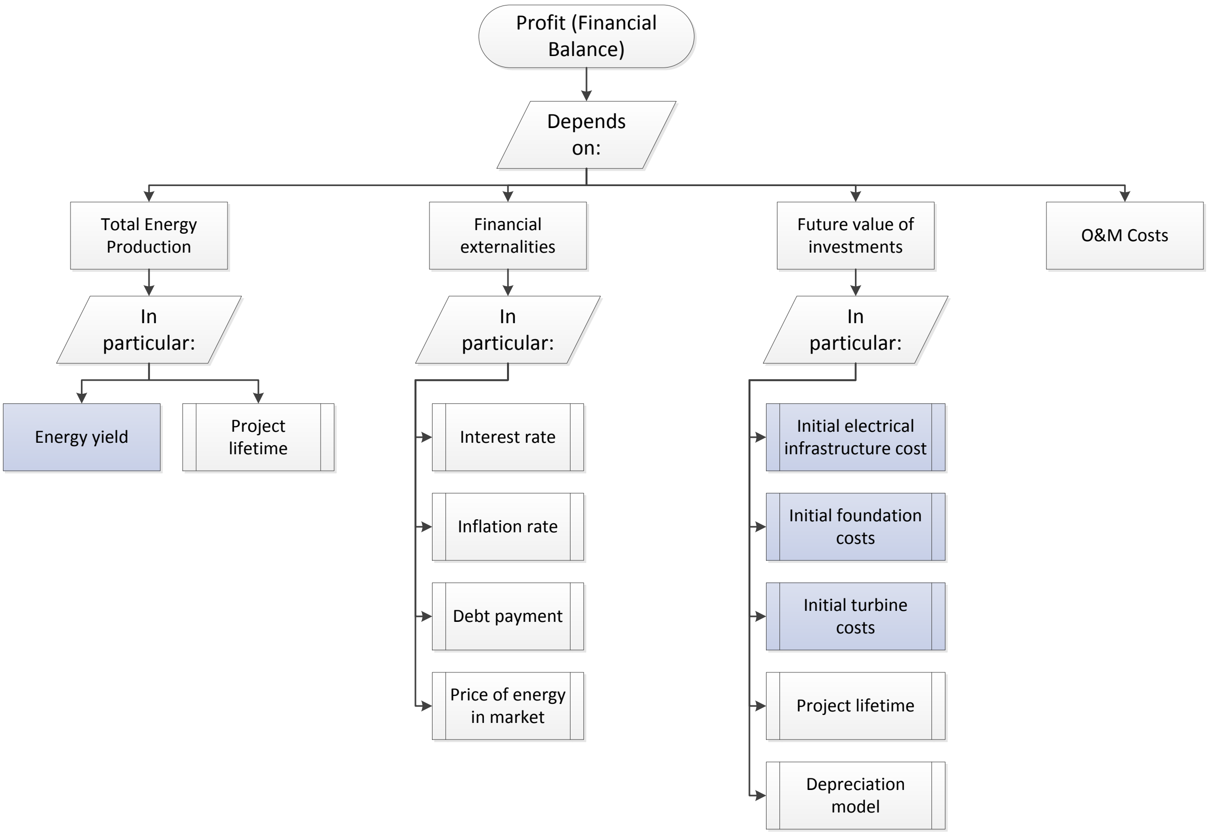


Figure 24. Profit function breakdown, as defined by Réthoré et.al (40).

The profit function considers several aspects of project valuation also present in the COE evaluation: financial parameters such as interest and inflation rates are contemplated by the annuity factor as well as the effect of lifetime on costs and profitability. This function considers depreciation of project assets as a burden to the overall profit, therefore generating a need to include an appropriate depreciation strategy that reflects not only the estimated state of the system and its projected current market value but also achieving concordance with tax regulations.

Regarding layout-related variables the energy yield reappears, which was analyzed in section 4.6.1. Three cost-related components which have also been treated before are present, thus no additional layout-related functional requirements will be obtained from the profit function analysis.

4.6.4. Net Present Value breakdown

The Net Present Value (NPV) is a well-known and understood function that can and is often used as an objective for all kinds of investment projects due to its “general” character. The advantage of using NPV as an objective function is that its terms are flexible, meaning that it is highly customizable and it can be as complicated as the designers wants it to by adding terms depending on how much information is available regarding the financial status of the project. NPV is based on simple cash flow discounting, translating periodic investments and profits to the present value of money, hence the name of the function.

The function breakdown here presented is what the author of this project considers to be the minimum requirements to achieve an objective function as comprehensive as those presented previously. Once again investment and operation and maintenance costs appear as extensions of the diagram.

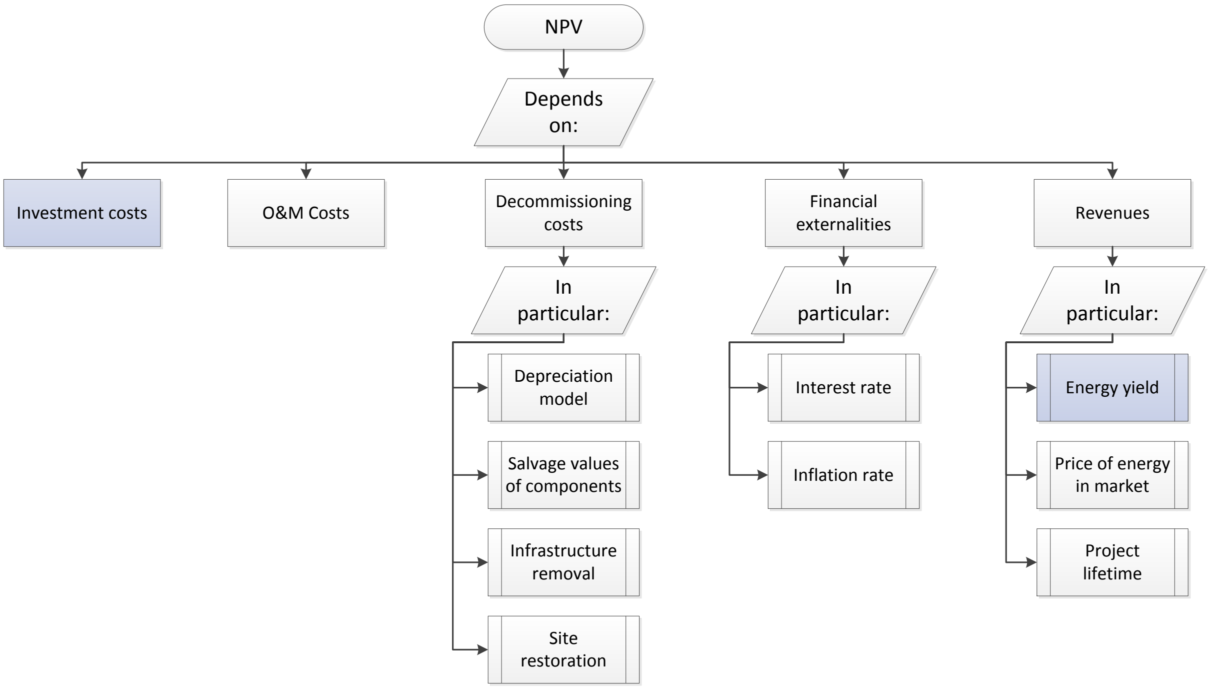


Figure 25. NPV function breakdown diagram.

The choice of NPV as an objective function is very straightforward and has the advantage that it is an indicator widely understood by most people with basic knowledge in project valuation. It also makes it possible for an investor to directly compare the benefit of building a wind farm against other project prospects.

The two components that are layout-dependent in the NPV objective function have also appeared in previous analysis, thus, no additional requirements can be obtained from it either.

4.7. Brainstorming

The final method of requirement extrapolation presented here is an informal one, since it consists of a personal reflection of the author of this project on the previous analysis outcomes and literature study. By brainstorming it is possible to provide some additional insight on missing requirements, reinforcement of the importance of some previously identified ones or reshaping the way they were originally stated. The results of this brainstorm are here synthesized and expanded from their original form:

- The objective function is the core of any optimization routine, therefore its definition is a very important step in order to obtain relevant results. The literature study revealed that authors often stress the need to develop more comprehensive objective functions that consider only the relevant problem variables (in this case layout optimization). The author of this project would like to express that research activities are now shifting their focus towards developing algorithms, routines and attaining better solutions to the problem. While this is perfectly fine (these developments target a requirement that needs to be fulfilled) few sources aim to develop new objective functions specially designed for the layout optimization alone, which is why most implementations rely exclusively on energy yield and COE.
- As an experienced software user the author cannot emphasize enough how important it is to have a streamlined user interface. When designing GUIs, extremes should be avoided: too simple and many functions are hidden or lost (and the perceived value of the program is reduced), too complicated and the user will feel overwhelmed by the tool. Adding graphical aids to the interface is often perceived as an improvement, either in the inputs or in the outputs. Plotting capabilities are also often overlooked, yet they can be especially helpful in sorting engineering problems. For the layout optimization problem the use of maps and other inputs from GIS software could greatly improve the user experience.
- Although not explicitly stated by any of the software publishers in their specification sheets a built-in help module or interactive user manual is a very valuable addition to any program, especially when used by users with mixed backgrounds and experience levels with computational tools. Engineering software often comes with a theory manual and a user manual, which often can be opened from the program itself. This is therefore added as a requirement.

- The analysis of uncertainty is a very important part of overall project assessment that the author feels is somewhat overlooked or not well documented for the case of wind farm energy yield. This functionality would be greatly appreciated by investors and project developers.
- Integrated support structure design in wind farm planning tools is now commonplace, yet it is unclear for most programs if only monopile design (which are the most widely used structures to date) are supported. The design process of jacket structures is considerably more complex since they are constituted of complicated three-dimensional trusses that make load determination more difficult. However, jacket structures can be preferred to monopiles (depends on site conditions) and thus providing the user with the option to choose between more than one support structure would add versatility to computational tools.
- A common way to define wind conditions is by means of the Weibull distribution. If the form and shape parameters are known, as well as the mean wind speed, then the generated distribution will accurately represent realistic site conditions. However a problem arises when the parameters are unknown or the user would like to generate the distribution from measured wind data. This function is provided by some of the commercial tools analyzed, and the author considers that it is a very important feature to include in any wind farm design tool.
- Many tools are capable of determining loads on the support structure, yet a function that only EMD WindPro has and would be worth extending to more practical applications is checking for compliance with the relevant standards (in particular, the IEC-61400).
- Providing the user with a catalogue of wind turbines to choose from is a very good idea; however, as was the case with automated standard verification this feature is so far only included in EMD WindPro.
- While performing the breakdown of objective functions a recurring thought was related to the financial parameters used for their evaluation, in particular interest and inflation rates. It is obvious that they must be somehow included in any tool that used COE, profit or NPV as an objective function, yet the source of those parameters is unknown. Allowing the user to use custom, up-to-date values would result in better financial appraisals.

4.8. Shaping and prioritizing functional requirements

Now that all possible methodologies of functional requirement extrapolation were applied all the observations and conclusions must be gathered, filtered and combined to arrive at the final list of user requirements. However, in order to properly define what particular problem will be tackled within the framework of this project the requirements must be prioritized by applying relevant criteria. Three criteria were chosen for this purpose, which are:

- Relevance (*R*): if a particular requirement or problem variable appears with high frequency in the analysis methodologies used it can be interpreted as a highly relevant aspect to address, due to the recurrence of the requirement. Some others are also highly relevant due to their addressing being a pre-requirement for further development. Relevance is also assigned from the point of view of an offshore wind farm developer, meaning that variables and requirements significant to the onshore layout problem only will score a lower relevance.
- Feasibility (*F*): this indicator takes into account both the nature of the requirement and the ability to implement the requirement given the current state of the art of both knowledge and tools available. It also encompasses the difficulty associated with said implementation.
- Pertinence (*P*): the final indicator aims to reflect how important and achievable is the implementation of the requirement from the perspective of the current project, time wise, scope wise and considering the current capabilities of the MZ tool.

All the identified requirements must be shaped into concrete actions, strategies and possible developments that may constitute a real engineering problem. Many of the identified problems/requirements are too broad or abstract as formulated in the previous sections; therefore some of them were reshaped, reinterpreted or combined to arrive at a total of 56 identified functional requirements. Ten categories (types) of requirements were identified, defined as described in Table 11.

Table 11. Classification of functional requirements.

Requirement Type	Description
A	Software requirements and inputs
B	Layout related requirements
C	Modeling and calculations requirements
D	Graphics and user interface
E	Environmental aspects
F	Optimization routine requirements
G	Financial aspects
H	Support structures
I	Electrical aspects

J	Miscellaneous

Some remarks regarding the classification:

- Category A contains those requirements tied to the performance of the program itself, as well as additional options and special inputs.
- Category J, named "miscellaneous" contains those requirements whose characteristics don't match with any of the other categories.
- Many requirements involve models; for example, assessing impacts on wildlife requires a model to achieve this goal. However, since this is a problem more closely related to environmental issues, it is classified as type E rather than type C. Category C therefore contains modeling aspects that are not mainly related to any of the other requirement types.

Also presented for every requirement an indicator denominated source (*S*), referencing the extrapolation methodology used to obtain them. Sources are classified as follows:

Table 12. Categories of the source indicator (*S*).

Source Code (S)	Methodology Used
I	Software outputs
II	Scientific literature
III	Layouts and constraints
IV	Objective functions
V	Brainstorming

The requirements per category and their respective scores for each of the selected indicators are presented in Appendix A: Lists of Functional Requirements in the form of tables. The 56 functional requirements are distributed as shown in Figure 26.

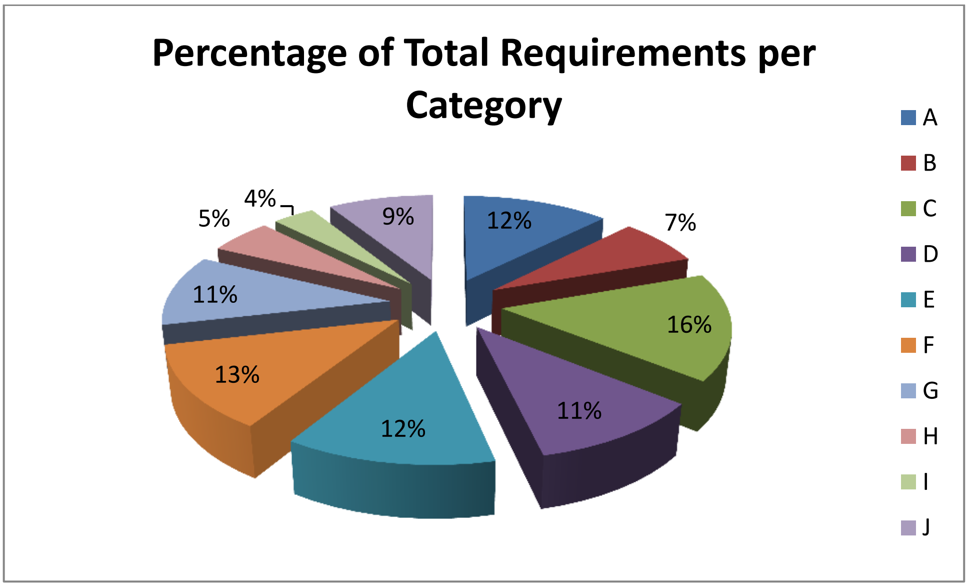


Figure 26. Pie chart representing percentage of the number of requirements per category to the total.

The priority indicator developed as part of this project serves as the main tool to assess the true importance of every functional requirement, not only in general but also to the present project. It is defined as the arithmetic addition of Relevance, Feasibility and Pertinence. Figure 27 presents the average value of priority per category, from highest to lowest. This average is computed using the individual priority scores of each item within the category. The categories with highest average priority also present lower standard deviations, indicating that there is a uniform trend for these types of requirements to consistently rank as very important within the context of wind farm design tools.

Layout-related functional requirements are ranked with the highest average priority, which is not surprising considering that the layout is influenced and influences practically all of the relevant variables to offshore wind farm optimization. Interestingly though, layout requirements constitute a quite small share of the total pool (only 7%), which only reinforces the importance that should be given to addressing them.

Discussing the lowest ranked category (type D, graphics and interface) also proves to be interesting, since the requirements in this group scored high in relevance due to reasons discussed previously on how graphical aids add a significant amount of value to computational tools. However, due to the difficulty of implementation of said solutions and the required knowledge and expertise on the field of GUI design these requirements scored low on feasibility and pertinence, leading to a poor performance in overall priority.

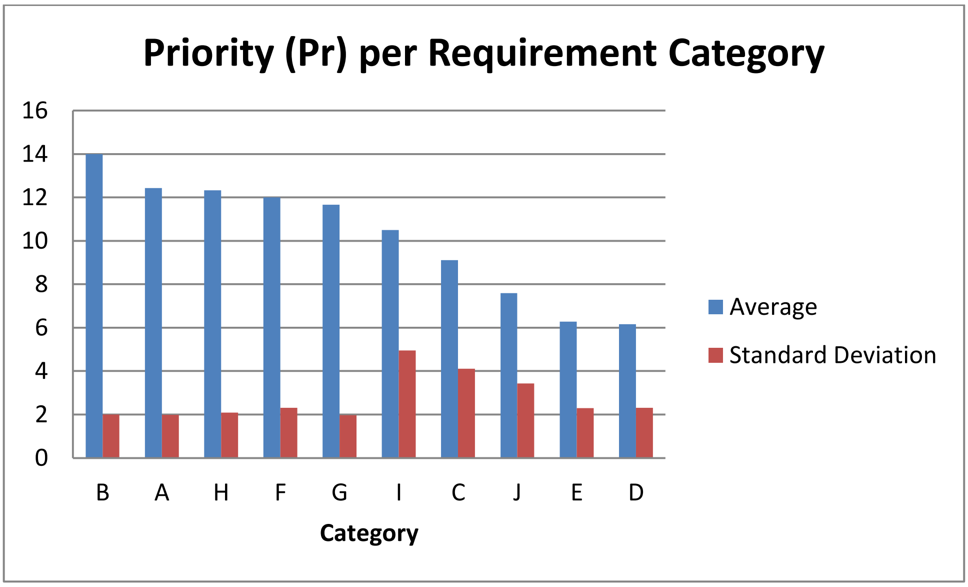


Figure 27. Average priority and standard deviation per functional requirement category.

In order to select the requirements that will be addressed by the present project only those with very high (above 13) priority will be considered. Table 13 presents the requirements that satisfy this condition.

Table 13. Functional requirements with highest priority.

No.	Requirement	Type	Pr
1	Achieve a high-quality solution in a reasonable amount of time.	A	15
2	Ability to either receive the power curve as input or otherwise generate an approximation to it.	A	15
3	Ability to generate optimized symmetrical (or slightly organized) turbine layouts.	B	15
4	Ability to generate optimized non-organized layouts.	B	15
5	Restrict available turbine siting area due to various constraints.	B	15
6	Apply a constraint of minimum turbine separation.	B	15
7	Include a robust wake analyst that can handle any layout.	C	15
8	Include COE/LPC as an objective function due to its known relevance to the problem, comprehensive formulation and acceptance by most users.	F	15
9	Include models to determine hydrodynamic and aerodynamic loads.	C	14
10	Ability to evaluate financial performance indicators, such as NPV or IRR for the project.	G	14
11	Include the option of defining custom financial parameters, such as interest and inflation rates.	G	14
12	Automated support structure design.	H	14

13	Implement an automated cable topology optimizer, closely tied to the layout optimization routine.	I	14
14	Allow the user to use create and use custom wind turbines.	A	13
15	Have a fully implemented and functional wind rose.	C	13
16	Give the user a choice of at least two objective functions.	F	13
17	Implement a heuristic optimization algorithm.	F	13
18	Implement a local search optimization algorithm	F	13
19	Ability to calculate loads on the structure and automatically determine compliance with the IEC-61400 standard.	H	13
20	Ability to analyze uncertainty of the energy yield calculation.	J	13

The following chapter aims to determine which of these requirements will be addressed within the framework of this project. The numbering of the requirements in Table 13 is also used for referencing purposes.

5. Identification of problem(s) to be addressed

5.1. Chapter Introduction

Table 13 presents a pool of requirements that, according to the structured analysis performed, have the highest priority for the present project. For purposes of requirement selection it may be considered that all of them have the same weight, in order to avoid choosing those that have the highest rankings in the table. The selection process must be smart and based on practical and pragmatic reasons, rather than by simple selection according to the scores. An analysis of every requirement in Table 13 is presented in this section, acting as a final funnel in order to shape the target problem.

5.2. Already treated problems

The first simple step is to discard all those requirements that have already been treated or implemented in the MZ tool, whose main features were described in section 3.

The tool already has an option for the user to input a discretized power curve; even though this approach may not be the most exact it is good enough considering the true purpose of the tool and the minimum accuracy required. Therefore, requirement number 2 may be considered satisfied to an acceptable level.

Regarding the wake analyst requirement (number 7) a previous project by González (42) concerned the implementation of a general purpose algorithm that could determine wake losses for any layout, in contrast to the previously existing routine which could only deal with symmetrical arrays. This algorithm was successfully implemented in the MZ tool and therefore this requirement has also been fulfilled.

The MZ tool is also capable of optimization using COE/LPC as an objective function; therefore requirement number 8 may also be discarded.

Requirement 14 has also been addressed to a certain extent. While no database of turbines that the user can choose from is implemented, the MZ tool does allow the user to define all relevant turbine parameters as inputs to the program. While a higher level of detail and specification could be achieved, the current state of the implementation of this requirement in the project context is considered satisfying enough and thus discarded.

Requirement number 1 is a special one due to its general nature and overall relevance to computational problem solving as a whole. Achieving high quality solutions in a reasonable amount of time is a target that may be applied to every step, routine and algorithm developed and thus it is implicitly included as a “must have” characteristic, which is also permanently addressed by the developer of the code. Therefore, it is discarded from the list, not because it has no importance to the problem but because its relevance is already implicit in all further developments.

5.3. Requirement clustering

Due to the nature of many of the requirements listed in Table 13 it is possible to group many of them into clusters, which tackle problems that are in one way or another interconnected and thus could provide a “package” that constitute a full problem definition. Three clusters were identified from the list, which shall now be treated in detail.

5.3.1. Cluster A: Layout exclusive

Cluster A is composed of requirements 3, 4, 5, 13, 17 and 18. This group is the largest one of the three, and contains all those requirements focused exclusively on optimization of the layout itself (turbine siting issues only). The need to be able to design and optimize both symmetrical and unsymmetrical layouts was clearly determined by several of the requirement extrapolation methods used, which means that they constitute the most important issues to be addressed within this cluster.

Requirement 13, concerned with optimization of the cable topology, is one that must be tied to any routine that optimizes non-symmetrical layouts. For symmetrical layouts determining the best cable route (and thus the optimal cabling distance) is not extremely challenging due to the stiffness and uniformity of the solutions; it can almost be said that for square layouts the cable routing is almost predefined. However, when dealing with irregular layouts the ever-changing turbine positions makes the task of writing an algorithm that is versatile and flexible enough to find the optimal cabling route extremely challenging, yet necessary for an optimization that accounts for all relevant problem variables, being cabling distance one of the most important ones.

Total cable cost depends not only on cabling distance but also on the type and size of cable used. Several cable models exist to predict cable size based on transmitted power, however, it is well known (and intuitive) that larger power outputs require larger, more expensive cables. Therefore, as the number of turbines per branch increases the cable design needs to be adapted to support the infrastructure and properly reflecting the added burden on the total cable cost. In reality, cables with different diameters compose a single branch, with the caliber reducing as the distance from the substation increases. The cabling topology of the Anholt offshore wind farm in Denmark (Figure 28) is a good example of this phenomenon, in which three different cable sizes are used due to the higher power concentrations closer to the substation. Devising a way of including this variable tariff based on number of turbines in the optimization brings the solution closer to reality.

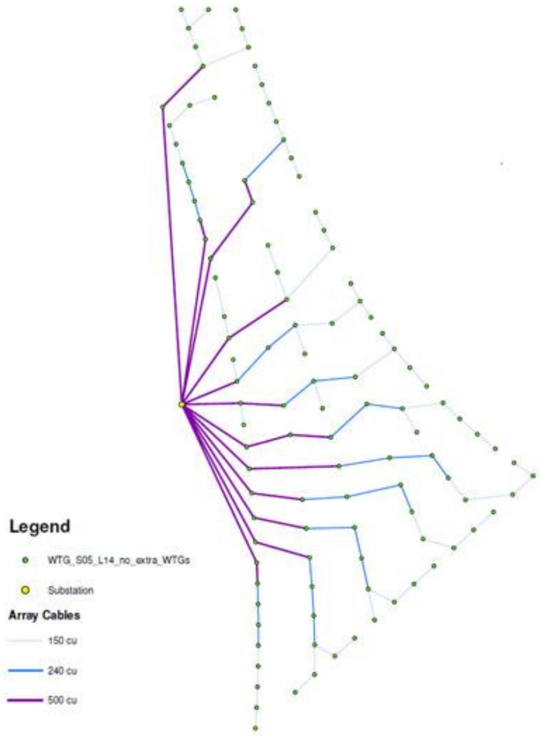


Figure 28. Anholt offshore wind farm cable design, Denmark. Reproduced from (56).

Requirement 5, which embodies the recurrent area restriction problem, constitutes a great challenge depending on the scope given to it. Consider the case presented in Figure 29.

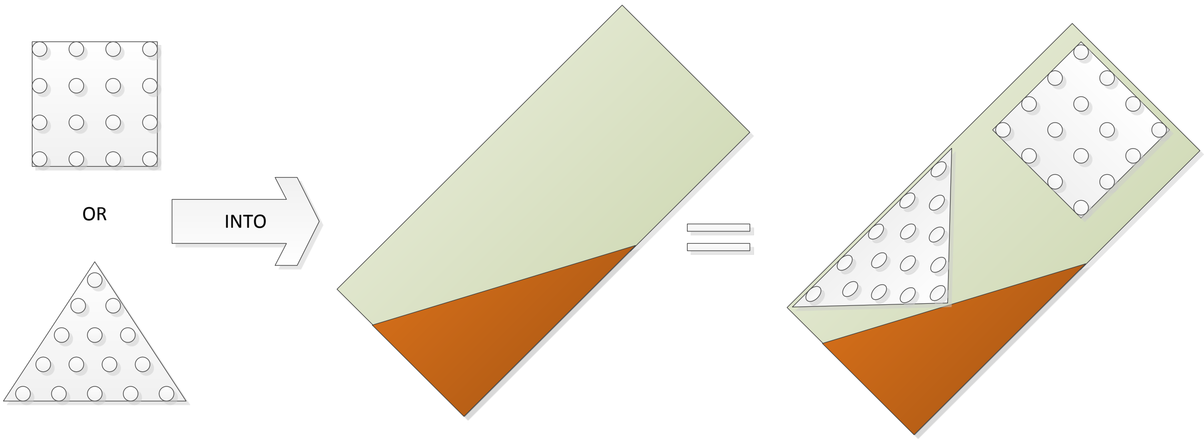


Figure 29. Placement of symmetric layout in an allowed area with restrictions (orange).

The figure illustrates the problem of placing a symmetric layout wind farm inside a target area with a restriction zone. This issue is very straightforward to solve, since the stiff geometry of the wind farm allows for the reduction of the problem to "placing area 1 inside area 2", or just finding a way to fit one geometric shape inside another. The area defined by the wind farm may be expanded/contracted and rotated (as Figure 29 illustrates) until the condition is achieved, effectively turning the area restriction problem into a geometrical and mathematical problem. More importantly, the cable topology is unaffected if the problem is treated from this scope due to the unchanged turbine positions, as was discussed previously. The fact that the cable topology optimization is disjointed from the area restriction and turbine siting problem indicates that this is not an ideal approach.

If a non-symmetrical layout is used the scenario is completely different, as shown in Figure 30.

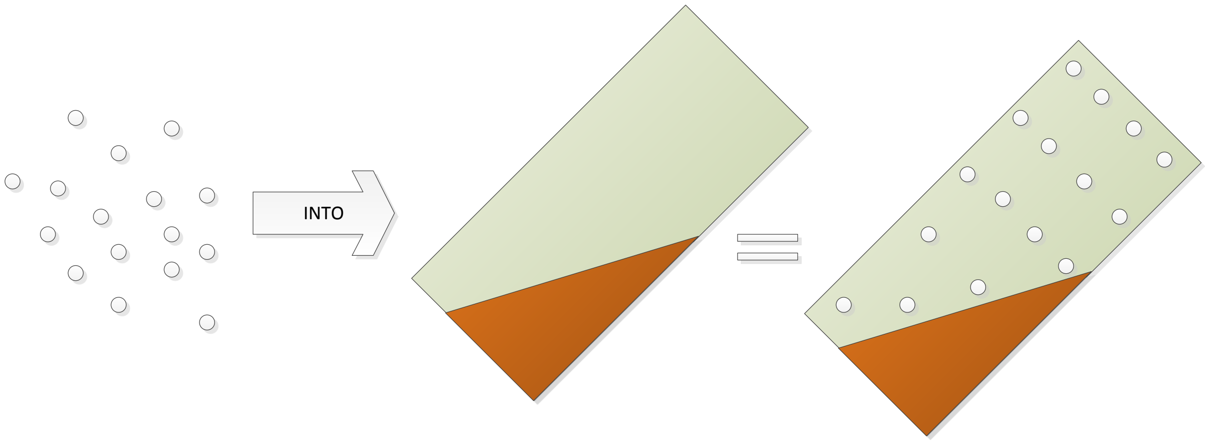


Figure 30. Area restriction problem for a non-symmetric final layout.

In this case the problem can no longer be reduced to "fitting area 1 inside area 2", quite simply because there is no clearly defined area 1. By allowing some randomness in the final solution turbine siting becomes an individual problem, and thus the stiffness of the layout is eliminated. Cable topology becomes not only a relevant but a challenging problem in this case, since there is no longer a "prescribed optimum" as the one that existed for exclusively symmetric layouts. If the cabling topology optimization issue is solved for random layouts then adding the area restriction problem becomes once again a constraint, since each individual turbine is able to determine its own ideal siting position while accounting for the effect of not only neighboring turbines but also the effect on cabling distance while being "forbidden" to sit in certain areas.

The final two requirements of cluster A (17 and 18) deal with the technicalities of the optimization routine itself, more specifically, which method will be used to solve the problem. When attempting to find the solution to an optimization problem the choice of approach is heavily guided by the problem definition itself.

5.3.2. Cluster B: support structures

Cluster B includes requirements 9, 12 and 19. These requirements are all related to support structure design, which depending on the approach used may or may not have an effect on the layout. This is particularly true for requirement 12, especially if the functionality of the MZ tool is extended to allow for optimization and use of more than one type of support structure. If this were the case then it may occur that due to varying water depth conditions and a freedom of support structure choice leads to certain turbines being moved closer to shore or further away, again considering that variables such as cabling distance and wake losses are also considered by the main optimization routine. This idea can be synthesized as follows: if support structure design is included as part of the problem, rather than an outcome of the calculation, then the problem becomes not only more challenging but also capable of affecting the wind farm's layout.

Requirement 9, dealing with modeling of conditions and loads can be viewed as one that can be made as complicated as the designer wishes to. While current models to assess hydrodynamic and aerodynamic loads are considered sufficient for pragmatic structure design, there is ongoing research on how to improve these modeling techniques in order to yield more accurate results, in a manner analogous to the wake modeling issue.

One clear example of current modeling issues and their effect on the layout is pointed out by (57), regarding wave conditions. When modeling wind, inflow turbulence models are already well known and have been validated extensively due to their use in the now-mature onshore wind industry; however, the standard practice for wave modeling is to represent the irregular (random) waves using a linear wave. This approach, while being very simple is too unrealistic and thus hydrodynamic loads are not determined with a high degree of accuracy and thus the support structure cannot be truly optimized since the real conditions are not properly represented. Ongoing research is concerned with improving and refining wave modeling, in particular using second-order models that offer a better representation of the randomness behind the natural phenomena.

Requirement 19 can be classified as partially treated by the MZ tool, since the tool is able to determine the loads in the structure for a limited amount of load cases although subjected to sizable simplifications that leave room for improvement. The load cases currently evaluated are:

- Operation at rated wind speed, maximum wave in one-year extreme sea state.
- Parked turbine, reduced gust in 50-year average wind speed, maximum wave in 50-year extreme sea state.
- Parked turbine, maximum gust in 50-year average wind speed, reduce wave in 0-year extreme sea state.

Safety factors and external conditions are defined as dictated by the IEC 61400-3 standard, just as the functional requirement identified demanded. The optimization routine uses an initial guess for the tower top diameter and the length of the transition piece; all other geometric parameters of the support structure are determined by the routine.

As was already mentioned the tool currently has the option of optimizing for a monopile structure. If a possibility existed to be able to design for more than one support structure type then options such as determining not only the best type of foundation but also its optimal design become available, further reinforcing the tool's capabilities.

In conclusion, requirements included in cluster B can be considered interesting and relevant for the scope and extent of this project only if they are approached from a perspective in which they are no longer additional calculations but can have an effect on the layout itself, effectively becoming "translatable" into working variables for the optimization.

5.3.3. Cluster C: profit driven requirements

The final cluster is comprised of requirements 10, 11 and 18, all of which are in one way or another related to financial aspects of the project. Analogous to the previous clusters, requirements here analyzed should be treated from the perspective of the possible effect that their assessment could have on the layout, which is not very straightforward for financial parameters and considerations.

Requirement 10 involves the calculation of cash flows and revenues to obtain financial performance indicators, and it may be either a post-processing option (for final feasibility evaluation) or as part of the layout optimization if it directly affects the objective function. The parameters used for this evaluation are those mentioned in requirement 11. This constitutes the more traditional view of project valuation: the project's main characteristics and overall design is obtained, translated to costs and forecasts of future revenues enable the calculation of yearly cash flows. Then, VPN or IRR calculation becomes very straightforward.

In section 2.3 an overview of most objective functions was presented, in which NPV and profit were identified as possible maximization targets. From this perspective the layout may be subject to alterations, especially if the use of more than one support structure and integration of an all-purpose cabling topology optimization routine is enabled. In short, aiming for a wind farm that maximizes profits throughout the complete life time of the project will probably produce a final layout that differs from that designed using a different objective function. Even if profit maximization does not constitute the target of the optimization routine, making it part of the multi-objective approach will surely have an effect on the outcome.

Requirement 18 is included in this cluster only because it can be easily associated with profit driven optimization. Requirement extrapolation methods from existing software tools revealed that often the choice between energy yield and a cost-oriented objective function (LPC, profit, VPN, etc.) is provided to the user. Nevertheless, using energy yield as an objective is not a very comprehensive approach to solving the layout problem since all cost-related variables and restrictions are left out of the scope of the solution. Furthermore, yield can be seen as a calculation step towards determining a more comprehensive and complex project indicator, such as LPC, which means that energy yield cannot be placed in the same level of significance as other objective functions. Only if cost-oriented objective functions are given to the user as options it makes sense to include more than one of them since they would constitute two different approaches providing results that offer similar complexity and reveal detailed design results.

5.4. Other loose requirements

In this section all requirements from the original table that have not been clustered or discarded are analyzed, in order to determine if they should be a part of the final problem formulation.

Requirement 6, while initially appearing as very straightforward has actually some physical related problems and considerations that can make it hard to tackle. When performing a literature study on layout optimization it is not uncommon to find authors that recommend that the separation distance between turbines is within a certain range, and not exceed or be below certain thresholds. These recommendations have two problems:

- Often they do not coincide, since they are mostly derived from observation or experimental results. For example, Bierbooms (58) suggests the spacing of offshore wind turbines should be above 6D (with only extreme exceptions being allowed under this value), Mittal uses a minimum separation of 5D (24), Pérez et.al (6), Kusiak et.al

(25) and Wan et.al (32) set this minimum at 4D and Réthoré et.al (40) go as far as setting a minimum in his work at 1D.

- Turbine separation is a variable that can have two effects on project LPC: if the turbines are too close together the cabling distance is minimized, but the wake losses are increased and a subsequent power loss is observed. If the turbines are too far apart, wake losses are minimized but cabling distance and cost increase. Therefore it is easy to imagine that a curve of LPC versus spacing will have one or more minima, being the lowest point in the curve the optimal separation distance. Since LPC is a project-specific variable this curve will also be project-specific, making it difficult to generalize where the optimal separation could be; in an extreme case there could be a project in which turbine and site characteristics make it optimal to place turbines with a 4D separation.

Some background behind the existence of a minimum separation distance can be found in literature. Pérez et.al (6) do provide the reason behind the existence of a minimum distance: “the wake model selected (Jensen) is known to provide appropriate results for distances higher than four rotor diameters, and for safety reasons, the minimum distance between turbines within the wind farm is limited to four rotor diameters”. Kusiak et.al (25), who also use a minimum separation of 4D use as an argument that “ensuring sufficient spacing between adjacent turbines reduces interactions, e.g., wind turbulence, thus diminishing the hazardous loads on the turbine”. Kusiak et.al further elaborate on this spacing constraint by tracing it back to (46), where the minimum distance of 4D is based on certain domain heuristics, as shown in Figure 31, where array losses remain relatively constant from unidirectional wind until 4D. The authors mention the possibility of replacing this assumption with more complex constraints, such as setting the minimum distance between two turbines as a function of wind direction and probability (related to the second problem identified for the requirement).

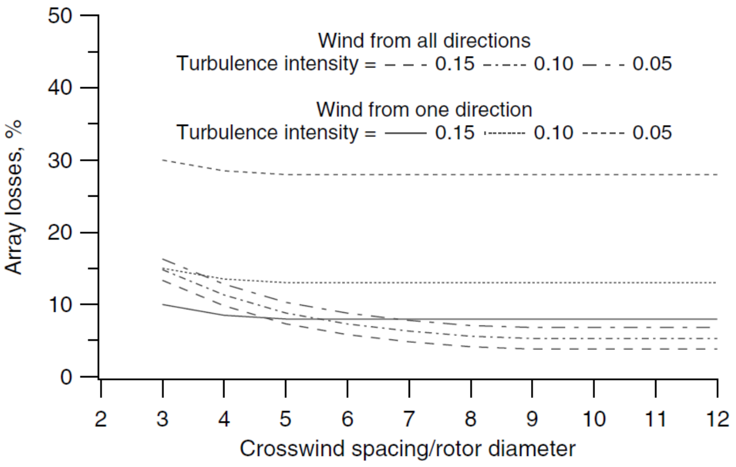


Figure 31. Wind farm array losses as a function of crosswind turbine spacing. Reproduced from (46).

Another explanation is that offered by Réthoré et.al (40), who mention that standards (not specified) require larger than 2D separation distances due to the detrimental effects of turbulence on the downwind turbine for shorter spacing. What can be concluded from this various attempts at justifying the minimum separation distance is that there is still some research required that points out if treating the problem as a constraints with a fixed range is the best approach to include it in a layout optimization routine. Modeling problems and restrictions do not constitute a fully valid justification for the use of the minimum separation distance since the origin of the constraint is not from a physical derivation but rather from a method limitation, which may be solved investing some time in research and development of a more appropriate model. However, the effects of additional turbulence created by a turbine siting too close is a real concern that does explain the need to draw a line from which turbine siting should be maybe not completely restricted, but appropriately flagged as potentially detrimental.

Requirement 15 is a particular one since there are many ways that a designer may achieve the goal of "implementing a fully functional wind rose". This requirement achieved a score of 5 in relevance due to the importance of having average wind speeds and probability for a detailed assessment of energy yield and to account for the effects of directionality in wind farm performance. On the other hand, the assessment of feasibility and pertinence resulted in a score of 4 for both indicators, placing the requirement on the lower end of the high-scoring priority list. This is due to the fact that, while implementing a discretized wind rose via numerical inputs for a fixed number of sampling directions is not a highly challenging task, standard wind rose visualization may require an upgrade of the program's graphical capabilities.

Finally, requirement 20 had a high priority score, stemmed by the overall importance that wind farm developers and program designers place on determination of best and worst case energy yields. Yield uncertainty is mainly associated with three sub-problems: the inability to properly determine, model and characterize site conditions (site uncertainty), errors associated with properly obtaining the power curve and projected availability uncertainty. In order to address this particular requirement a great deal of research is therefore needed. The problem becomes even more complex if cost uncertainties are introduced in the problem definition, especially those related to the design of the support structure, since regarding the improvement of the design of these structures, "there are still many uncertainties that question the design requirements used so far" (59).

5.5. Requirement selection and final problem definition

Now that all the top ranked requirements listed in Table 13 have been grouped, analyzed and discarded when appropriate the final problem to be addressed by this particular project may be defined. The problem funnel diagram presented in Figure 32 is a useful metaphorical representation of the process followed by the author to arrive to this point.

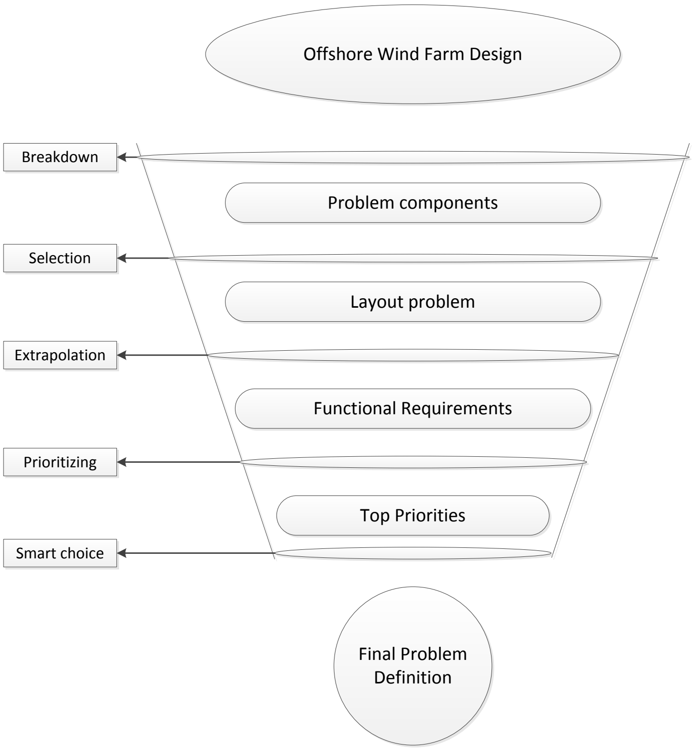


Figure 32. Funnel diagram illustrating the problem identification and narrowing procedure.

Following from the filtering process of the requirements with highest priority presented in sections 5.2 through 5.4 it may be concluded that cluster A constitutes an excellent base for the problem definition, due to the following reasons:

- On average, all layout-related constraints consistently achieved the highest priority scores.
- All requirements included in cluster A are closely related to each other and some of them (for example layout optimization, routine implementation and cabling optimization) need to be addressed simultaneously and in parallel to achieve a satisfactory solution. In short, they form a nice “package of objectives”.
- The requirements in this cluster correspond well to the initial expectations of how the problem definition would be shaped after a systematic approach was used.
- The implementation of said requirements is challenging, yet corresponds well to the extent of the project and the timeframe in which it is enclosed.

The problem definition is therefore an amalgamation of the requirements in the cluster, and is defined as follows: “design, implement and test a series of computational solutions within an existing offshore wind farm design tool that enable it to produce organized and non-organized layouts while respecting area restrictions and realistic/optimal cable topologies”. The following chapters deal with the methods and solutions developed to address this global problem.

6. Developing Solutions: Cabling Topology

6.1. Introduction and Methodology

The optimization of the electrical layout is a logical first step in the development of solutions, due to the fact that the cable layout must be defined before any objective function can be evaluated; it is therefore a prerequisite to addressing the siting problem. This chapter presents the methodology used to characterize the cabling optimization by representing it as an existing topological problem and then apply a solution strategy designed to solve such a problem.

6.2. Problem identification

In section 5.3.1 the problem of determining the optimal cable layout was introduced, alongside its characteristics and constraints. Due to the restriction of having no more than a maximum number of turbines per cable branch a solution that contemplates multiple cables stemming from a base node must be designed. This problem is usually tackled from the perspective of graph theory and optimal routing, which attempts to generate connections of varying nature over a set of nodes following preset rules and constraints. Four topology problems from graph theory resemble the cabling routing challenge:

- The traveling salesman problem (TSP), and its more generalized version, the multiple traveling salesman problem (mTSP).
- The minimum spanning tree (MSP) problem.
- Shortest path problem (SPP).
- The open vehicle routing problem (OVRP).

All of these compose a family of topology problems called optimal routing problems. An analysis and comparison with the cable topology optimization problem revealed that the Open Vehicle Routing Problem (OVRP) is the one that more closely resembles the characteristics and constraints involved in tracing a wind farm cable layout. The OVRP will therefore be treated here in more detail; for more information on the mTSP, MSP and SPP references (60), (61), (62) and (63) are recommended.

6.2.1. The OVRP

The Vehicle Routing Problem (VRP) consists in defining a set of optimal routes departing from a certain number of depots (starting points) to service a certain number of customers. It can be considered a special case of the mTSP, yet the reason why it is treated separately is because at the end of a vehicle/salesman’s route he does not have to return to the depot, hence dubbed “open”. Bauer et.al (63) have documented a successful application of this problem model and solution to the design of cable topology for offshore wind farms. The problem definition in their approach is as follows: “Given turbine and transformer positions and a cable capacity in number of turbines, and a set of cable routes minimizing the total cable cost, connecting every turbine to a transformer, not exceeding cable capacity, and such that cables do not cross each other”. In their work, they document the development of a solution algorithm to the problem and test it with real OWF layouts, some of which have more than one substation (depot). The results of their work are highly satisfactory and thus it may be a good option to treat the problem from this perspective.

The equations that formulate the VRP are nearly identical to those that constitute a traditional TSP, however, the constraints to the problem are altered and a set of so-called planarity constraints. To solve the linear programming problem that the formulation generates, Bauer et.al resort to the Clarke and Wright savings heuristic, which is one of the best known heuristic solutions to the VPR problem. Furthermore, branching and cable types are both acknowledged as being implementable in their solution, although the current version of the algorithm does not support these characteristics. More importantly, their work demonstrates that even though the problem formulation of the VRP contemplates the vehicles returning to the origin, it is possible to modify or interpret results in such a way that the algorithm determines routes and branches and not tours. The variation of the problem definition that includes this condition is named the Open Vehicle Routing Problem (OVRP), due to the “open” ends that the branches represent as opposed to closed loops stemming from a central point.

6.3. Solution Methodology

In order to provide a solution algorithm for the OVRP the work by Bauer et.al (63) will be central in the development of an implementable solution. In their work, they developed two algorithms based on the Clarke and Wright savings approach (dubbed planar savings heuristics, or POS) and an add-on to improve suboptimal routes that any of the POS routines may generate, named RouteOpt.

The POS heuristics differs from the original Clarke and Wright formulation in the following aspects:

- Allows the definition of open routes, that is, with a starting point in a depot (in the cabling routing problem this is the OHVS) and ending in a turbine without the additional condition of returning to the depot.
- Two arcs are no longer allowed to cross each other. While in reality cable overlapping may be solved using special connections during installation the situation is usually avoided due to the additional costs and risks associated with such a setup. The POS heuristics add an additional constraint (called the planarity constraint) to work around this problem by forbidding routes from overlapping.

The solution algorithms (POS1, POS2, POS1+RouteOpt, POS2+RouteOpt and best of POS) presented by Bauer et.al were tested and their performances were compared. In this listing the algorithms are ranked from most basic to most refined. As expected, more refined versions of the heuristic were more efficient and accurate but are also considerably more complicated and do generate an additional computational burden. However, the test instances revealed that POS1 generates cable routings that are only 5% more expensive that the known optimal solutions. Considering that cable routing is only one of the problems to be addressed in this project and not its main goal POS1 was chosen as the blueprint for the algorithm to be developed, in order to avoid the additional complexities of using POS2 or a combination of both. If the need arises, RouteOpt may also be added to the algorithm developed in order to further improve its outputs.

A guided example that illustrates how POS1 works is presented in Appendix C.

6.4. The Algorithm

Following the pseudo-code presented by Bauer et.al an algorithm was designed to implement the POS1. The algorithm is presented in the form of flow of logic diagrams. In order to simplify the presentation of said diagrams many processes have been grouped into subroutines, which are included in Appendix B; therefore, only the main routine is presented here. The POS1 algorithm finds optimal routes as follows:

a. Using the positions of every node (turbines) as inputs a cost matrix *C* is calculated, representing the distance of traveling from one node to another.
b. The initial routing is defined as all nodes connected to the central point (the OHVS) with a single arc (cable).
c. A savings matrix *S* is calculated from the cost matrix applying the definition of saving developed by Clarke and Wright (and slightly modified by Bauer at.al). While the

equation is not presented here, it is worth noting that a saving represents the cost that is spared by replacing the original arc (from the node to the central point) with a new arc connecting two nodes.

d. The elements in the savings matrix are sorted from highest to lowest saving.
e. The possible arcs that are defined by the savings matrix are then assessed individually to determine if their inclusion in the final routing is beneficial or not (in terms of reducing cost). This assessment consists in verifying that a single arc satisfies five conditions that deem it beneficial for the final solution. These conditions are discussed in more detail in the following section.
f. If the arc satisfies the five conditions, it is added to the routing vector *R* (containing the information of how to connect the nodes) and to its respective path (a single path represent a cable, connecting a certain number of turbines to the OHVS).
g. Steps 5 and 6 are repeated until no all arcs have been evaluated.

One noticeable change developed for the implementation here presented is that the savings matrix *S* has been replaced with a savings vector *S* serving the same purpose; this change is exclusively due to the fact that a vector is easier to sort than a matrix.

Figure 33 illustrates the symbol convention that all subsequent flow of logic diagrams will use in the present document. Figure 34 illustrates the POS1 algorithm to be implemented.

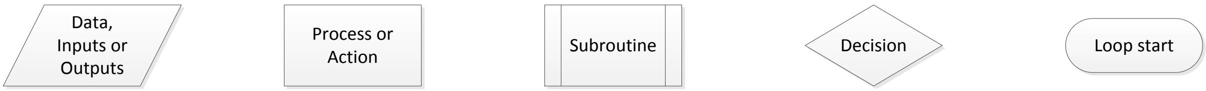


Figure 33. Symbol convention for flow diagrams.

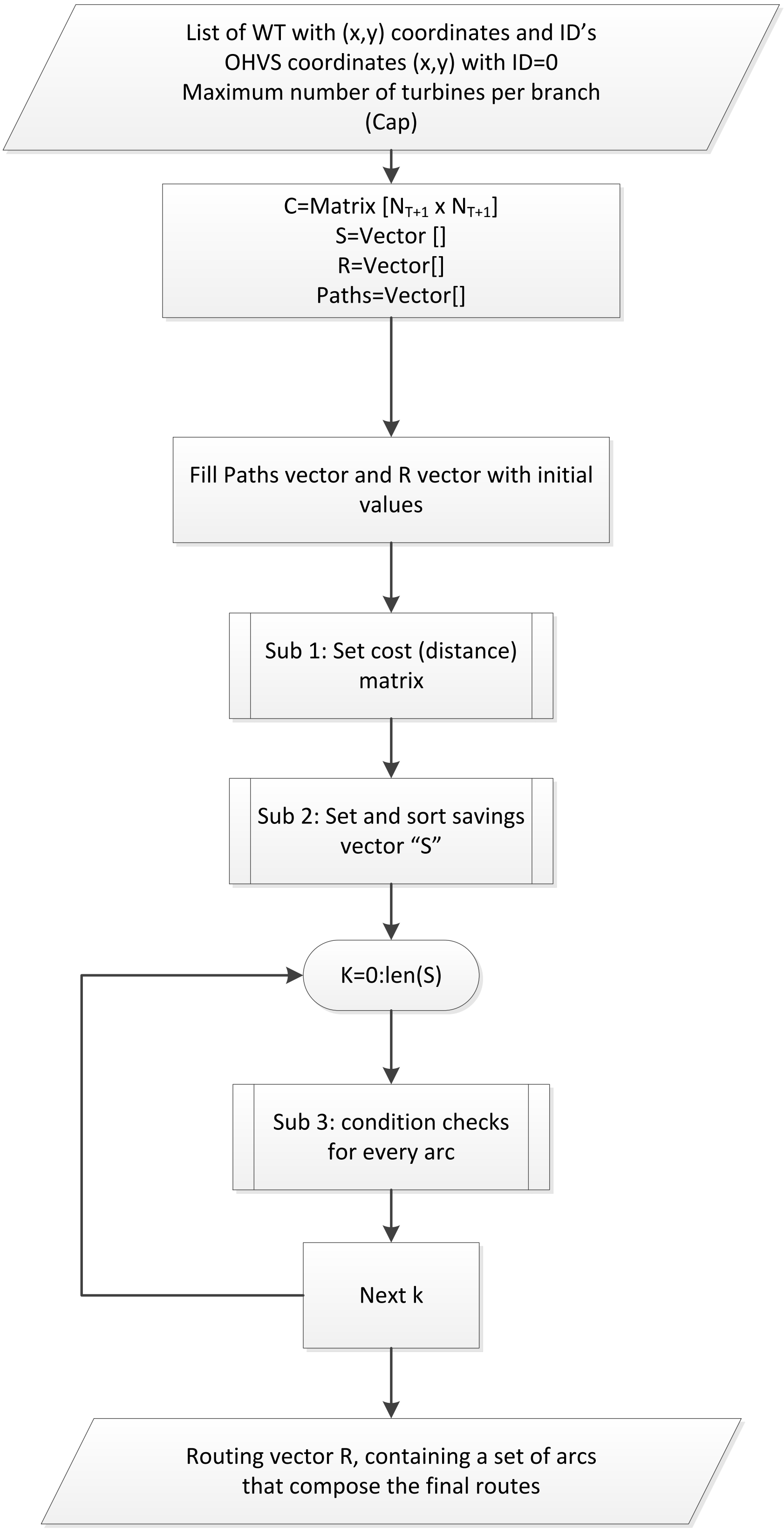

List of WT with (x,y) coordinates and ID's
OHVS coordinates (x,y) with ID=0
Maximum number of turbines per branch
(Cap)
C=Matrix [N_{T+1} x N_{T+1}]
S=Vector []
R=Vector[]
Paths=Vector[]
Fill Paths vector and R vector with initial values
Sub 1: Set cost (distance) matrix
Sub 2: Set and sort savings vector "S"
K=0:len(S)
Sub 3: condition checks for every arc
Next k
Routing vector R, containing a set of arcs that compose the final routes

Figure 34. Cable routing algorithm POS1.

The designed algorithm to implement POS1 contains three different subroutines, tasked as follows:

- Subroutine 1 is concerned with initializing and calculating the cost matrix “C”, which for the cabling routing problem is filled with distances between all possible pairs of turbines.
- Subroutine 2 calculates the savings for every possible arc and places the values in the savings vector “S”.
- Subroutine 3 is by far the most complex, since it is tasked with performing the five condition checks required to merge an arc into an existing route. The complexity of some of this condition checks implies that subroutine 3 has its own subroutines; a breakdown of such conditions is presented in section 6.4.1.

The algorithms final output is a new routing vector *R*, containing all arcs that construct the optimal cable routing. The paths vector which is initialized in conjunction with *R* contains the same arcs included in *R*, but organized per route. An example routing vector is given by Equation 20, with each tuple of values *(k,u)* representing the turbine ID’s that are connected by a single cable. The ID marked “0” is reserved for the OHVS.

6.4.1. Condition checks to add a new arc

In order to determine whether or not the addition of a new arc results in a better cable routing it must meet five requirements here presented (as conditions). The author considers that an appropriate breakdown that explains how each condition is evaluated is relevant, since some of them constitute subroutines of their own which are also included in Appendix B. The check of these five conditions is achieved by subroutine 3 in the proposed implementation.

Condition 1: Check if two nodes (k,u) are both in the same route or path

Consider the scenario in which an arbitrary routing vector has been modified by several arc mergers, resulting in:

$$R = [(2,0), (3,0), (4,0), (5,1), (1,2)]$$

Equation 13.

This is not the final routing vector, but rather an intermediate one being modified by the main loop of the algorithm. Now consider that arc (2,5) is to be evaluated against the five conditions. Nodes 1, 2 and 5 are already part of the same route, thus adding arc (2,5) would result in a closed loop which is undesirable, therefore, this arc must be rejected due to its composing nodes (k,u) being already in the same route. In order to reject said arc the paths vector may be used, since all individual routes are contained within it.

The compliance of this condition is implemented in subroutine 4.

Condition 2: Check if arc (k,0) is included in the routing vector R

This condition requires no special method or in-depth description since it only consists in verifying if a particular element is included in an array.

Condition 3: Check if wind turbine with ID=u in the arc (k,u) has only one neighbor

This check is required to avoid establishing an arc connecting to a node that is already a “bridge” between two other turbines or a turbine and the OHVS (in other words, prevents branching). Subroutine 5 (Figure B 5) achieves the check of this condition by counting the number of times a certain node (in this case, *u*) is present in the arcs that constitute the routing vector *R*. If any node has more than one neighbor it will be present in more than one arc and thus the count of repetitions of that particular node will be greater than one.

Condition 4: Check if capacity is not exceeded in any of the paths containing k or u

This condition prevents the merger of two paths if the resulting route has a turbine count higher than the maximum capacity per branch. Although not a difficult check to perform it is complex enough to be assigned its own subroutine (Subroutine 6). Every path in the paths vector is checked to see if it contains either *k* or *u*. When a path that does contain them is identified, the turbines in said path are counted and compared to the capacity.

Condition 5: Check that arc (k,u) doesn’t intersect other arcs in the routing vector R

This condition is reduced to a geometrical problem concerning line segments, which are not exactly straightforward to solve. The solution developed is based on the fact that the interest is to find whether to line segments intersect or not; the point at which this intersection occurs is not of concern for the method and thus the problem solution is simplified. Consider the two lines to be evaluated are the current arc *(k,u)* and an arbitrary arc in R, *(v,w)*. The arc connecting nodes *k* and *u* shall be defined as the anchor, while nodes *v* and *w* will be evaluated against this arc. Consider the three test cases shown in Figure 35:

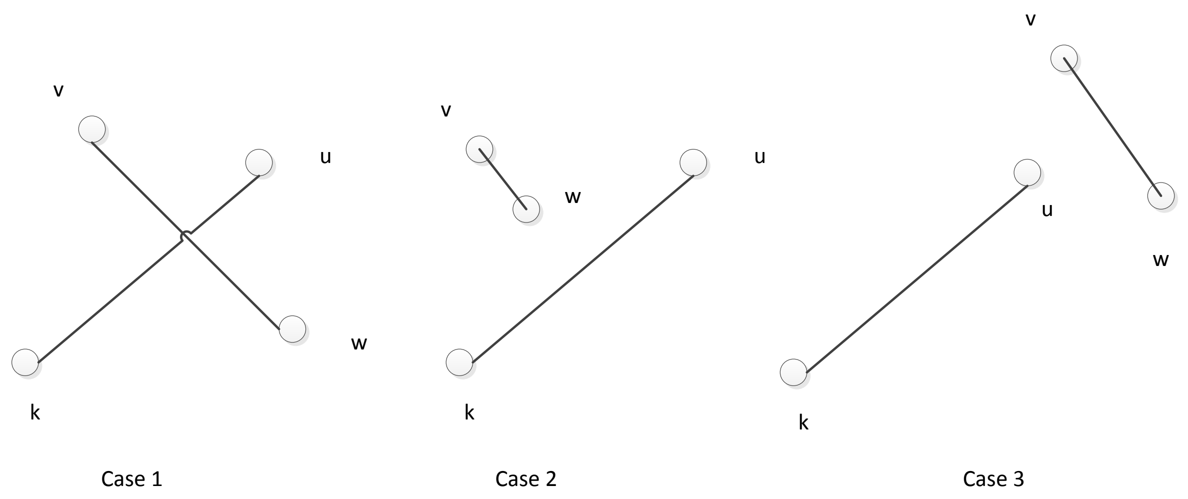


Figure 35. Test cases for Subroutine 7, checking for intersections.

Case 1 displays a simple intersection between the anchor segment *(k,u)* and the test segment *(v,w)*. Notice that in this case points *v* and *w* are in opposite sides of the anchor segment; this simple observation constitutes the base of Subroutine 7. Case 2 illustrates the opposite condition, in which both *v* and *w* are to "the left" of the anchor segment; clearly these segments do not intersect. Therefore, if points *v* and *w* are in the same side of the anchor line the test segment and anchor segment do not intersect.

Case 3 illustrates a special condition which must be considered within the algorithm. Test points C and D are to the left and right of the projection of anchor segment (A,B), yet the lines clearly do not intersect, thus failing the criteria previously proposed. Notice, however, that if the anchor line is changed to be (C,D) then the previous statement is true. Thus, any case may be evaluated using the following statement: "two line segments intersect if and only if the two nodes of one segment are on opposite sides of the other segment and vice-versa".

To verify whether a point is to the left or right of the anchor line the area of the triangle formed between the anchor segment and the point is calculated (using coordinates, not absolute lengths); thus the sign of this area will serve as an indication as to where the point lays in relation to the segment.

6.5. Results of Implementation

The POS1 algorithm was added to the MZ tool, contained by the layout analyst. The purpose of performing preliminary tests using the algorithm is not to determine whether the Clarke and Wright parallel savings algorithm is well-suited to solve the cable routing problem (this is treated in Bauer's work), but to verify that the implementation yields logical results and the routine behaves as expected. In order to do this, two different random wind farm layouts were used as inputs for the algorithm and the resulting cable topologies were analyzed to make sure that none of the imposed constraints to the solutions were violated and the calculated routing is close to what may seem an optimal solution.

The first wind farm layout and the resulting topology are shown in Figure 36. The maximum number of turbines per branch (capacity) was set to 3.

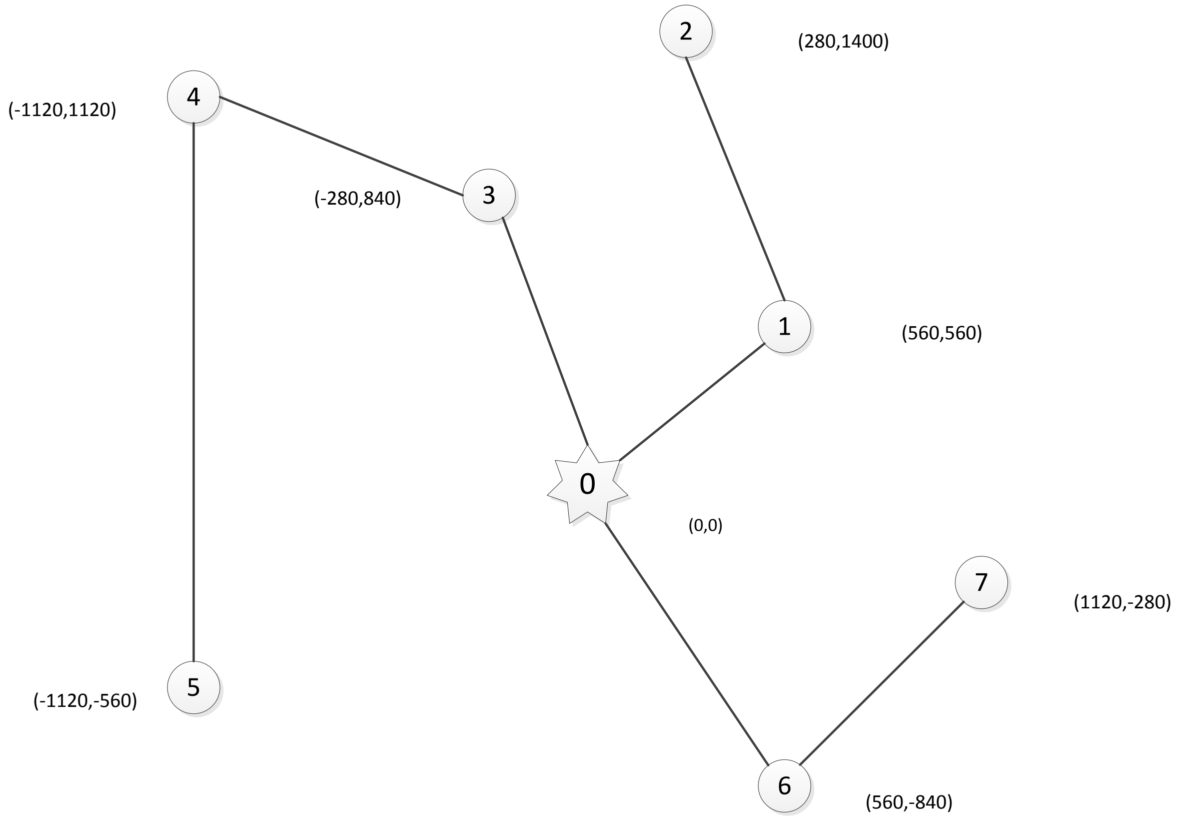


Figure 36. Test layout with seven turbines and resulting routing.

The test scenario using seven wind turbines arranged in a random layout yields good results, with all the constraints being satisfied. No branch exceeds the capacity and there is no cable intersection in the solution. A debugging process was used to follow the algorithm “step-by-step”, with the non-usable arcs being rejected according to one or more of the five conditions that constitute the heart of the POS1 algorithm. The only apparently non-optimal arc that appears in the final routing is (5,4), since it appears that it would make more sense to leave turbine 5 connected directly to the OHVS to obtain a shorter cable. These sub-optimal routes are mentioned by Bauer et al. (63) as a possible outcome of the POS1, which may be eliminated if RouteOpt is coupled to it.

A second layout was also tested; unlike the first one, a more traditional wind farm was used (that is, the turbine siting is more symmetric). The layout and the resulting cabling topology are presented in Figure 37.

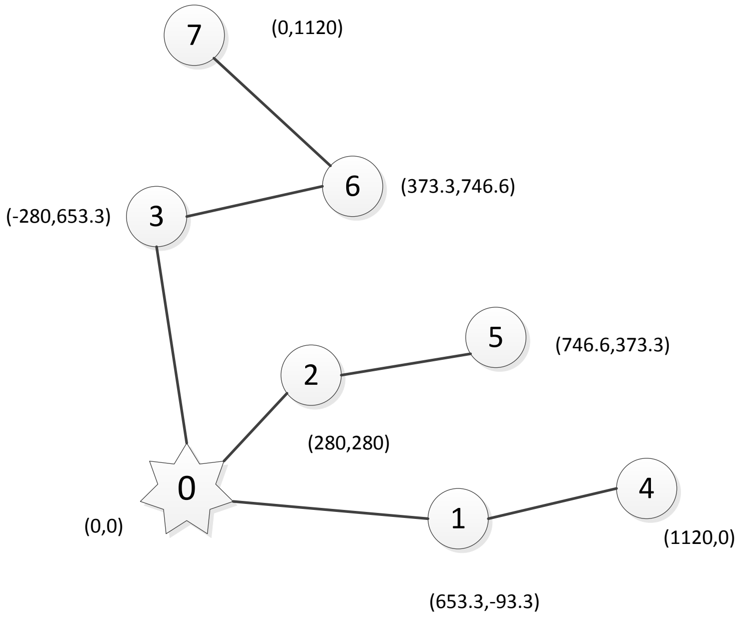


Figure 37. Test layout 2 with seven turbines and resulting routing.

The capacity was also set to three for this test case. The results show also a very "intuitive" routing, that once again satisfies all the problem restrictions. Moreover, none of the routes seem to be particularly sub-optimal for this layout.

From both tests it can be concluded that the implemented algorithm is performing as expected, and while it is possible that the resulting routings are not the exact optimal solution they are good enough for the purpose of this project. Bauer et.al mention a 5% deviation between the optimum and the outcomes of POS1; therefore, it is expected that most solutions obtained using this routine lie within that margin.

6.6. A Note on Objective Function Evaluation

Proper evaluation of the objective function is essential to the success of any optimization routine (as explained in Chapter 2); in the case of the MZ tool the objective function is the wind farm's LPC. The original MZ tool featured a grid-based default layout, with extensive use of symmetry for calculation simplification based on the fact that all wind turbines were organized in rows and columns. This kind of logic had to be completely removed from the MZ tool in order to enable it to calculate the LPC of any wind farm layout. One module that required special attention was the electricity analyst in which the infield and transmission cables are analyzed and their main properties and attributes recorded. This module required a complete overhaul due to the fact that for more realistic layouts stemming from non-grid symmetries each cable has a different length and thus the analysis can no longer use row-column logic and result duplication. This problem is better visualized in Figure 38.

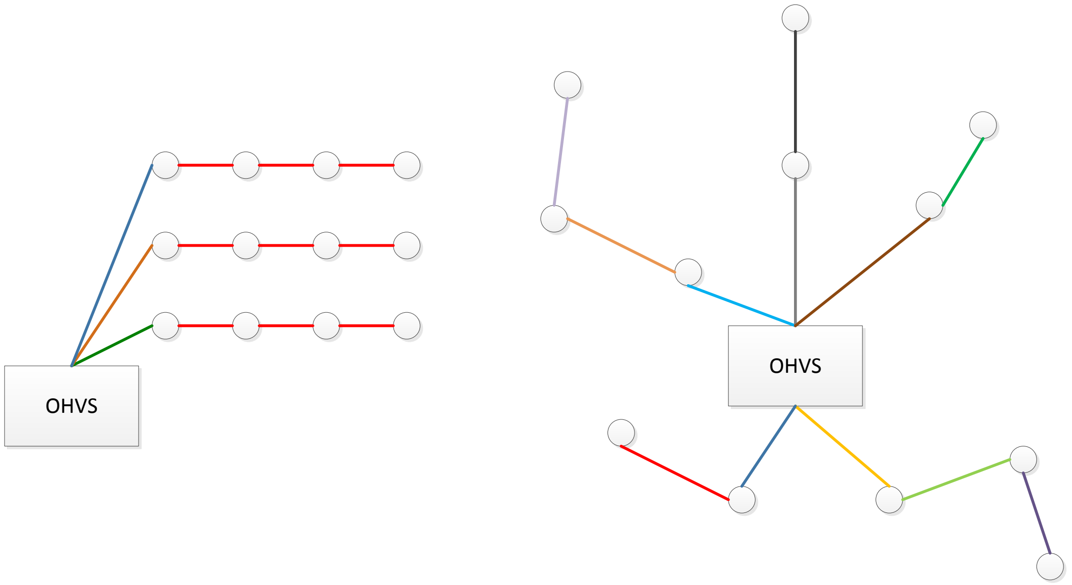


Figure 38. Cabling analysis in the original MZ tool (left) and the proposed modification (right).

In the original problem treatment symmetry enabled the design and analysis of a single infield cable (represented in red in Figure 38, left) since all of them have the same lengths and thus calculations for active and reactive currents, temperatures and losses were easily extended for the whole wind farm. Only the cables connecting each row of turbines to the platform were treated individually, due to the fact that the length of each of these cables varied according to the relative position of the row to the OHVS. The strategy used to modify the analyst can be roughly summarized as implementing a different way to generate and store cable properties due to the different geometry, while retaining the original electrical models.

One additional matter that needs to be addressed in order to have a “finished” algorithm (aside from a defined stop criterion which is treated in the following section) is the selection of a number of sectors in which the wind rose is divided for the wake analysis. In a previous work based on the same subject (69) it was determined that the performance of the wake analyst was a function of several parameters, with order of calculations having a linear increase with the number of sectors used to divide the wind rose to set the wind directions for the analysis (for example, using 12 sectors results in a division of the wind rose in 30 degree segments). Consider the schematic representation of a wake produced by an upwind turbine on a downwind one represented in Figure 39.

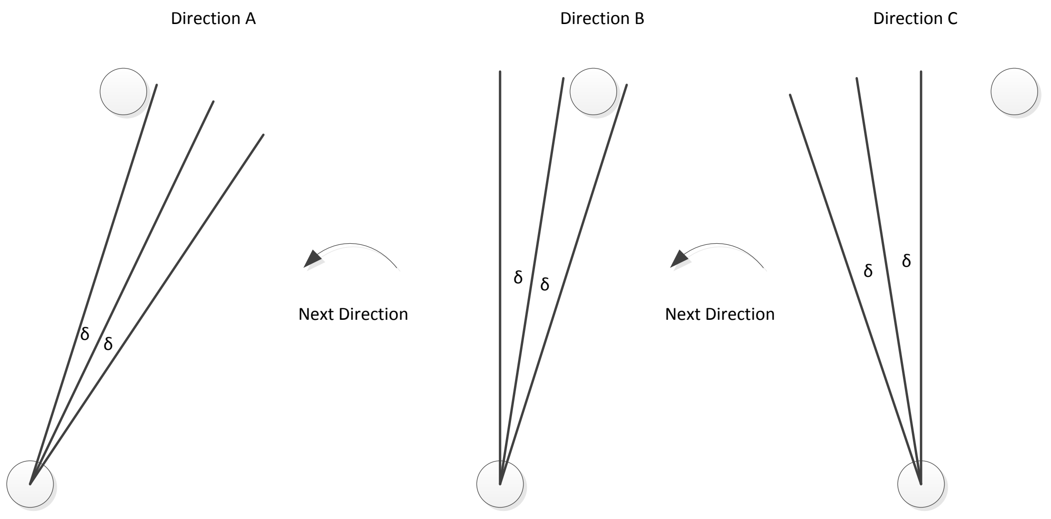


Figure 39. Schematic representation of a wake produced by a downwind turbine (bottom) on an upwind one (top) for three different wind directions and a "coarse" wind rose division.

This scheme represents a wake expansion from a turbine for three different wind directions. In this case the wind rose has been divided in "coarse" sections, which is equivalent to having a small number of wind direction sectors. What can be deduced from the figure is that for the first direction (direction A) the downwind turbine has no induced speed deficit from the upwind turbine; this also holds true for the last wind direction (direction C). The intermediate direction (direction B) has the downwind turbine completely immersed in the wake of the upwind one, thus generating a "full" wake effect. It is therefore very easy to deduce that the wake efficiency of this two turbine array will have a dependence on wind direction as shown in Figure 40.

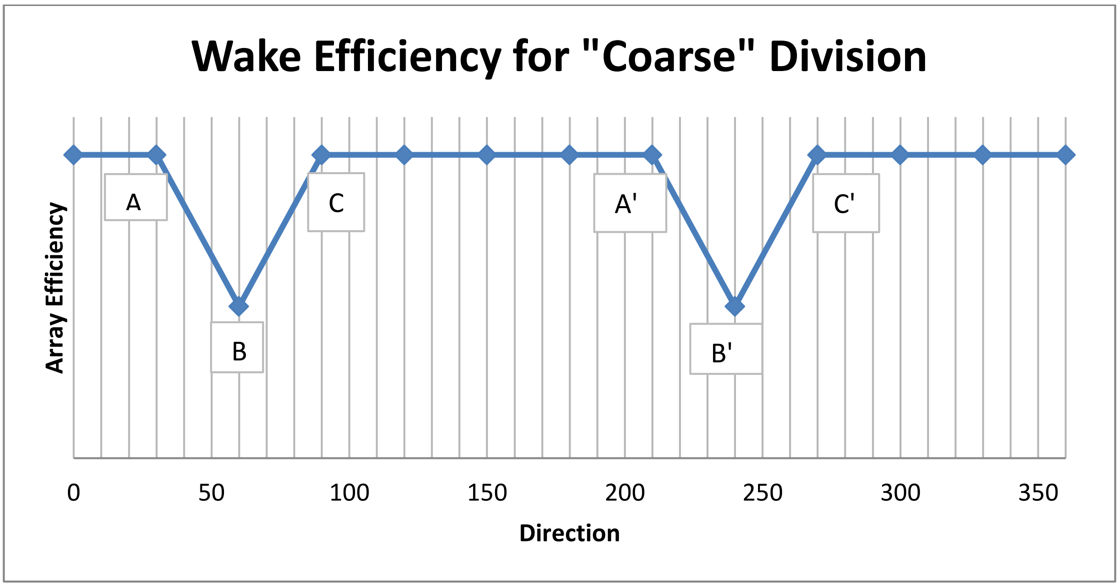


Figure 40. Wake efficiency for different wind directions, schematic plot. A 30° division of the wind rose was used, representing a “coarse” division situation.

The points A, B and C make reference to the directions used in Figure 39, whereas A’, B’ and C’ represent those same directions plus a 180° shift. The problem of using such a large angle step size becomes evident when compared with a situation in which a “finer” step size is used, such as the one depicted in Figure 41.

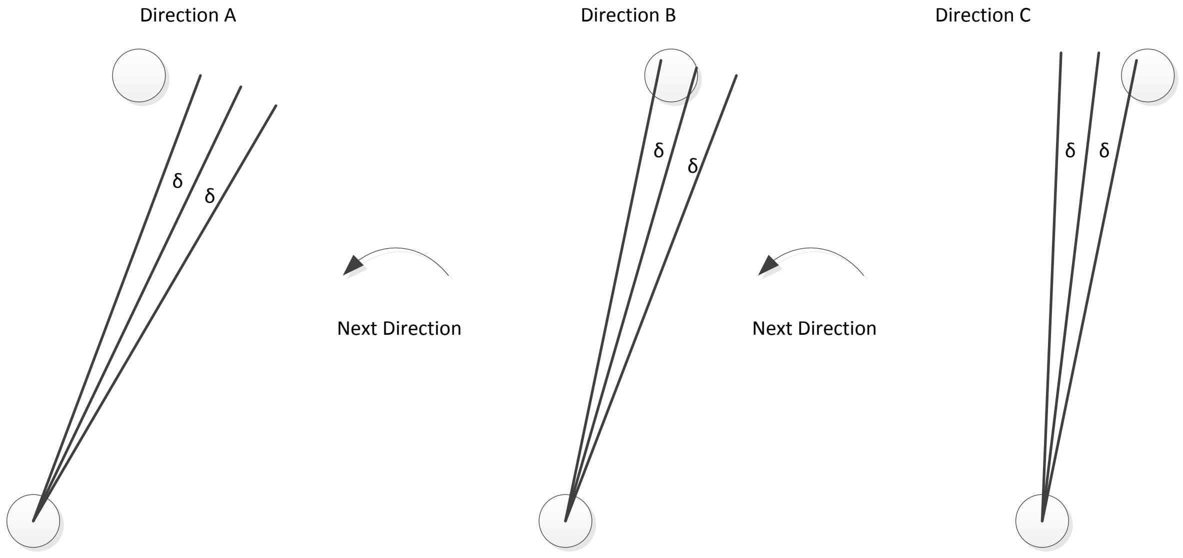


Figure 41. Schematic representation of a wake produced by a downwind turbine (bottom) on an upwind one (top) for three different wind directions and a “finer” wind rose division.

Using a “finer” division of the wind rose (more sectors) produces drastic changes in the results of the qualitative wake analysis. Direction A still reveals no wake effect of the upwind turbine on the downwind one. However, due to the smaller step size directions B and C now reveal a partial wake incidence on the downwind turbine. It is easy to imagine that the next direction sampled (direction D, not shown in the figure) will reveal an inexistent wake influence once again of one turbine over the other. Under these conditions the wake efficiency of the two turbine array as a function of direction will show a behavior as represented in Figure 42.

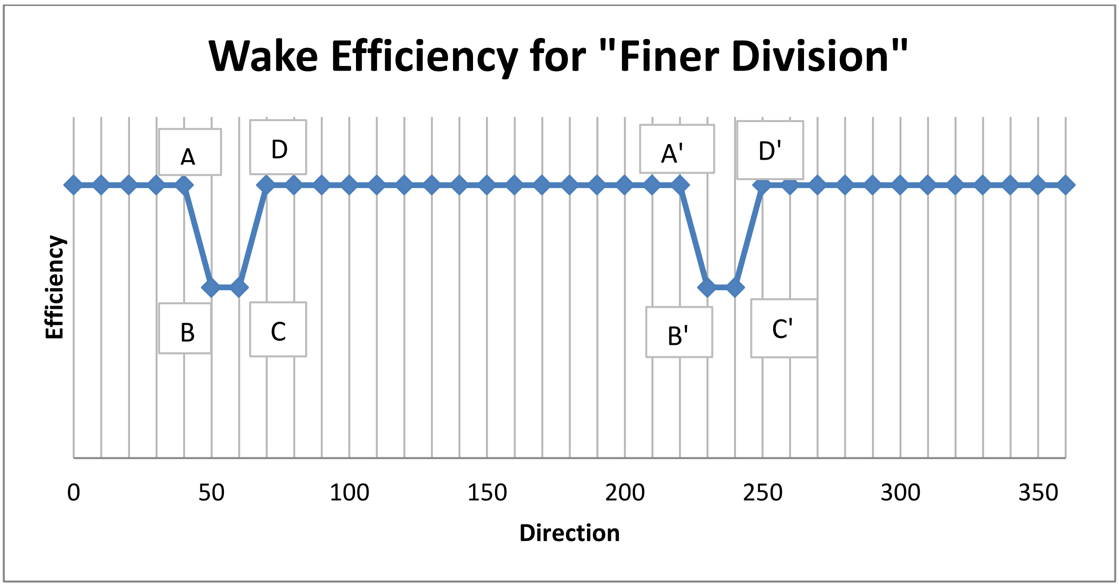


Figure 42. Wake efficiency for different wind directions, schematic plot. A 10° division of the wind rose was used, representing a "finer" division situation.

It becomes evident that the curve starts becoming "smoother" the finer the divisions are, since a greater number of partial wake effects are captured by the wake analyst.

Zaayer (2) justifies the use of angle step sizes no greater than 10° based on the same line of argumentation here presented; in his implementation 144 wind sectors (that is, 2.5° steps) are used to capture at least two partial wake effects on a rotor for a 5D separation between turbines. For the purpose of this project an angle step of 10° was chosen (to allow acceptable computation times while still capturing significant wake information), which is equivalent to 36 sectors.

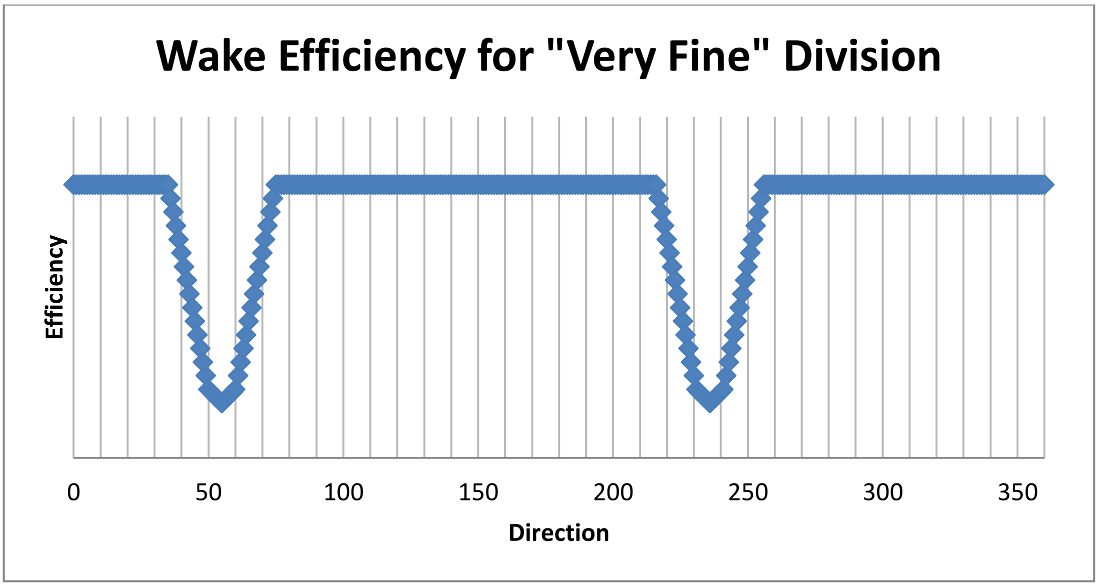


Figure 43. Wake efficiency for different wind directions, schematic plot. A 1° division of the wind rose was used, representing a "very fine" division situation.

7. Developing Solutions: Turbine Siting Problem

7.1. Introduction and Methodology

In order to generate an algorithm that is able to determine the optimal location of turbines over a continuous solutions space a suitable wake analyst and a cabling topology optimization routine were developed and implemented. The outcomes of this routines and analysts are required for the calculation of any objective function used for the offshore wind farm optimization algorithm. Now that these routines have been designed and implemented, the main problem to be addressed by this project will be introduced in detail.

In this chapter, a strategy to optimize the position of the turbines inside the wind farm area will be presented and an algorithm to implement said strategy will be developed and implemented.

7.2. The CMA-ES

In section 2.2 most of the optimization strategies that have been implemented to solve the turbine siting problem were briefly introduced and described. This review revealed that only heuristic approaches are interesting for the problem, due to the number of optimization variables and overall problem size typically associated with wind farm optimization, which makes finding exact solutions a computationally expensive process that not always yields a satisfactory result.

The CMA-ES approach is a heuristic evolutionary algorithm first proposed and described by Hansen (64) (65) which has recently gained momentum and overtaken genetic algorithms as the state-of-the-art solution for many multi-objective problems in engineering. The strategy is "a stochastic method for real-parameter (continuous domain) optimization of non-linear, non-convex functions" (64) that has already been successfully applied in at least one instance to the wind turbine siting problem by Rodrigues et.al (36).

The CMA-ES is a powerful heuristic that is able to overcome the most common problems associated with evolutionary strategies, namely:

- Poor performance (in terms of solution quality) optimizing badly scaled and/or highly non-separable objective functions.
- The need to use large population sizes to achieve convergence. When using CMA-ES, the population size can be chosen unrestrictedly due to the algorithms ability to “learn” through each subsequent generation and understand the problem better.
- Premature convergence of the population. This is avoided by the use of step-size control. However, as with any other heuristic routine, entrapment inside a local optimum may still occur.

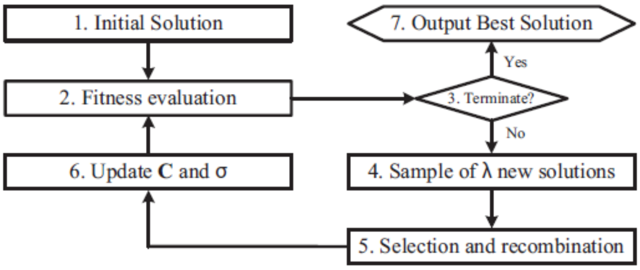


Figure 44 illustrates the flow of logic that characterizes the CMA-ES algorithm. The steps that comprise the routine are described in the subsections that follow.

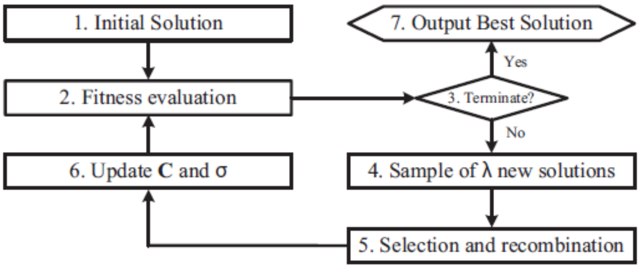


Figure 44. Flowchart for the CMA-ES algorithm, reproduced from (36).

7.2.1. Step 1: Defining an Initial Solution

In the initialization a solution to the problem (in this case, a layout) is randomly generated by sampling the turbine coordinates from a normal distribution. A solution to the turbine siting problem will therefore be a structure containing all the (x,y) pairs of coordinates for all the turbines in the layout. The problem dimension (search space dimension), denoted by *n* is therefore equal to two times the number of turbines, since each turbine has an *x* and *y* coordinate to be optimized.

7.2.2. Step 2: Fitness Evaluation

The randomly generated initial solution is tested in a process denominated “fitness evaluation”, which is essentially calculating the objective function (yield, LPC, profit, etc) for the given layout. The idea behind any evolutionary strategy is to maximize or minimize the objective function by finding the solution with the highest fitness, thus this evaluation needs to be performed every time solution is to be tested, as the figure depicts.

7.2.3. Step 3: Stop Criterion

Step 3 involves deciding whether or not to terminate the optimization loop by evaluating a stopping criterion that can be defined in a variety of ways. A convenient way to define this termination criterion is to limit the number of generations (solutions), which is analogous to restricting the maximum number of iterations that the main loop of the algorithm is allowed to run. Alternatively, stagnation of the objective function evaluation may also be used as a stop criterion. Nevertheless, there is no clear consensus so far of what the best stop criterion is for the CMA algorithm; for the purpose of this project some conclusions will be drawn on what an appropriate formulation for the criteria should be based on the convergence on the problem (section 7.3).

7.2.4. Step 4: Sampling of new solutions

If the loop is not terminated, a new set of solutions is obtained by sampling from a multivariate normal distribution with a mean *m*, standard deviation *σ* and covariance matrix *C*. Note that this distribution parameters are initially assigned recommended values and then updated after each CMA loop. This sampling procedure of λ new individuals obeys Equation 14.

$$x_k{}^{(g+1)} = m^{(g)} + \sigma^{(g)} N\left(0, C^{(g)}\right) \; for\; k = 1 \ldots \lambda$$

Equation 14.

where *N(0,C)* may be read as “sample from a normal distribution with mean 0 and covariance matrix *C*”. Subindex g stands for “generation”, meaning that new individuals are sampled using distribution parameters from the previous generation of solutions.

7.2.5. Step 5: Selection and Recombination

The new generation of solutions is then recombined, much like a regular genetic algorithm. However, unlike a genetic strategy in which all solutions are arbitrarily recombined, in the CMA-ES only half of the solutions (λ/2) with the best fitness are recombined and used to obtain a new mean for the sampling distribution. This selection and recombination process (step 5 of the algorithm) enhances the ability of the strategy to avoid entrapment in the local minima of the problem. The mean of the new distribution is calculated with Equation 15, which assigns weights to every solution (with the best ones having the highest weights).

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i x_{1:\lambda}{}^{(g+1)}$$

Equation 15.

Where μ is the number of selected recombination points (λ/2), w_i are the recombination weights and $x_{i:\lambda}{}^{(g+1)}$ g is the i$_{th}$ best individual out of generation (g+1).

7.2.6. Step 6: Update covariance matrix and step size

The last step of the routine is updating the covariance matrix *C* and the step size *σ*. In order to do both updates the concept of evolution paths must be introduced. “We call a sequence of successive steps, the strategy taken over a number of generations, an evolution path” (64). To construct the evolution path using the parameters from the previous generation Equation 16 is used.

$$p_c{}^{(g+1)} = (1 - c_c) p_c{}^{(g)} + \sqrt{c_c(2 - c_c)\mu_{eff}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}$$

Equation 16.

The internal parameter cc is the learning rate for the update of the covariance matrix, whose value can be found in the parameter table (Table 14). The parameter μ_{eff} is the variance effective selection mass, given as:

$$\mu_{eff} = \left(\sum_{i=1}^{\mu} w_i{}^2 \right)^{-1}$$

Equation 17.

The evolution path is then used to update the covariance matrix for every iteration, which is done according to Equation 18.

$$C^{(g+1)} = \left(1 - c_1 - c_\mu\right) C^{(g)} + c_1 p_c{}^{(g+1)} p_c{}^{(g+1)^T} + c_\mu \sum_{i=1}^{\mu} w_i y_{1:\lambda}{}^{(g+1)} \left(y_{1:\lambda}{}^{(g+1)}\right)^T$$

Equation 18.

The internal parameters used in this equation are c_1 (Equation 19), c_μ (Equation 20) and $y_{i:\lambda}{}^{(g+1)}$ (Equation 21).

$$c_1 = \frac{2}{n^2}$$

Equation 19.

$$c_\mu = \min\left(\frac{\mu_{eff}}{n^2}, 1 - c_1\right)$$

Equation 20.

$${y_{i:\lambda}}^{(g+1)} = \left({x_{i:\lambda}}^{(g+1)} - m^{(g)}\right)/\sigma^{(g)}$$

Equation 21.

In an analogous manner to the covariance matrix update, the standard deviation (step size) of the multivariate distribution is updated using the so called conjugate evolution path calculated using Equation 22.

$${p_\sigma}^{(g+1)} = (1 - c_c){p_\sigma}^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}\,{C^{(g)}}^{-\frac{1}{2}}\frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}$$

Equation 22.

Finally, the step size is updated as:

$$\sigma^{(g+1)} = \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|{p_\sigma}^{(g+1)}\right\|}{\left\|\sqrt{n}(1 - \frac{1}{4n} + \frac{1}{21n^2}\right\|} - 1\right)\right)$$

Equation 23.

The only parameter in Equation 23 that has not been introduced so far is d_σ, which is a damping parameter for the update of the standard deviation.

Once the covariance matrix and the step size have been updated a new fitness evaluation is performed for the new individuals and the process is looped until the stop criterion is met.

7.2.7. Default parameter values and overall algorithm

The previous subsections introduced the steps that compose the CMA-ES along with the required equations for an implementation of said strategy. This evolutionary strategy is heavily dependent on a set of internal parameters, many of which do not even appear in the main equations of the routine. The default values for all these parameters are summarized in Table 14.

Table 14. Default values for the internal parameters of the CMA-ES. Reproduced from (65).

Selection and Recombination:

$$\lambda = 4 + \lfloor 3 \ln n \rfloor, \quad \mu = \lfloor \lambda / 2 \rfloor \ ,$$

$$w_i = \frac{\ln(\mu + 1) - \ln i}{\sum_{j=1}^{\mu} (\ln(\mu + 1) - \ln j)} \quad \text{for } i = 1, \ldots, \mu \ ,$$

Step size control:

$$c_\sigma = \frac{\mu_{\mathrm{eff}} + 2}{n + \mu_{\mathrm{eff}} + 3}, \quad d_\sigma = 1 + 2 \max\left(0, \sqrt{\frac{\mu_{\mathrm{eff}} - 1}{n + 1}} - 1\right) + c_\sigma$$

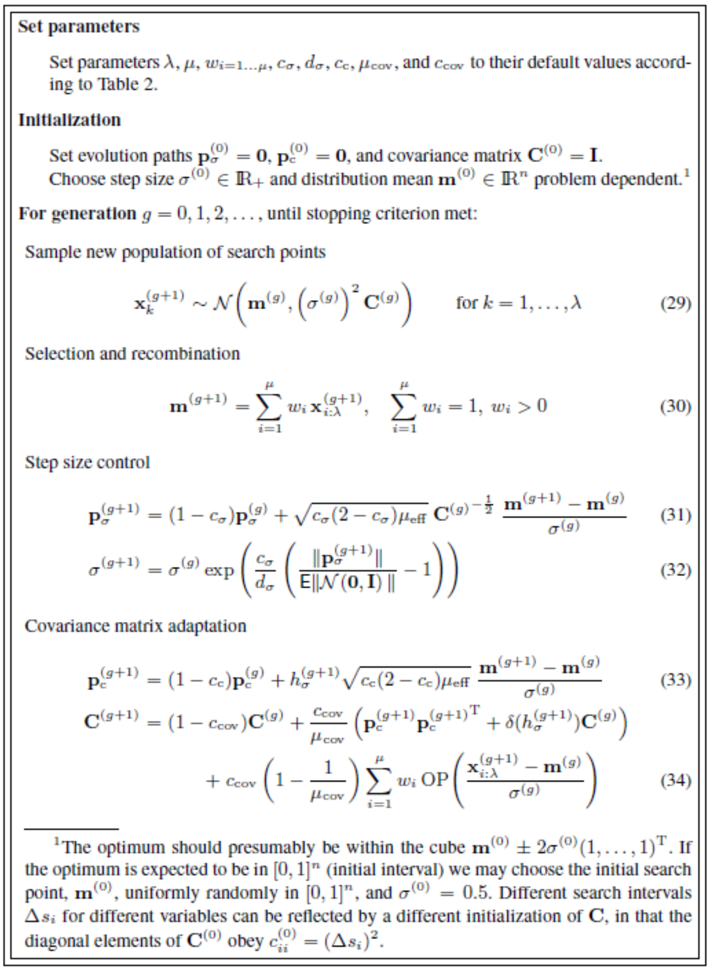
Covariance matrix adaptation:

$$c_{\mathrm{c}} = \frac{4}{n + 4}, \quad \mu_{\mathrm{cov}} = \mu_{\mathrm{eff}}$$

$$c_{\mathrm{cov}} = \frac{1}{\mu_{\mathrm{cov}}} \frac{2}{(n + \sqrt{2})^2} + \left(1 - \frac{1}{\mu_{\mathrm{cov}}}\right) \min\left(1, \frac{2\mu_{\mathrm{eff}} - 1}{(n + 2)^2 + \mu_{\mathrm{eff}}}\right)$$

In addition to these parameters, Rodrigues et.al (36) initialize the covariance matrix $C^{(0)}$ as the identity matrix, the evolution paths p_c=p_σ=0 and the standard deviation $\sigma^{(0)}$=0.5, which is consistent with the recommendations of Hansen (65). The synthesized algorithm for the CMA-ES strategy is presented in Table 15.

Table 15. CMA-ES synthesized algorithm. Reproduced from (65).



Set parameters

Set parameters λ, μ, $w_{i=1\ldots\mu}$, c_σ, d_σ, c_{c}, μ_{cov}, and c_{cov} to their default values according to Table 2.

Initialization

Set evolution paths $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$, $\mathbf{p}_{\mathrm{c}}^{(0)} = \mathbf{0}$, and covariance matrix $\mathbf{C}^{(0)} = \mathbf{I}$.
Choose step size $\sigma^{(0)} \in \mathbb{R}_+$ and distribution mean $\mathbf{m}^{(0)} \in \mathbb{R}^n$ problem dependent.[1]

For generation $g = 0, 1, 2, \ldots$, until stopping criterion met:

Sample new population of search points

$$\mathbf{x}_k^{(g+1)} \sim \mathcal{N}\left(\mathbf{m}^{(g)}, \left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}\right) \qquad \text{for } k = 1, \ldots, \lambda \tag{29}$$

Selection and recombination

$$\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \, \mathbf{x}_{i:\lambda}^{(g+1)}, \quad \sum_{i=1}^{\mu} w_i = 1, \; w_i > 0 \tag{30}$$

Step size control

$$\mathbf{p}_\sigma^{(g+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\mathrm{eff}}} \, \mathbf{C}^{(g)^{-\frac{1}{2}}} \, \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{31}$$

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{(g+1)}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right) \tag{32}$$

Covariance matrix adaptation

$$\mathbf{p}_{\mathrm{c}}^{(g+1)} = (1 - c_{\mathrm{c}})\mathbf{p}_{\mathrm{c}}^{(g)} + h_\sigma^{(g+1)} \sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})\mu_{\mathrm{eff}}} \, \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{33}$$

$$\begin{aligned} \mathbf{C}^{(g+1)} &= (1 - c_{\mathrm{cov}})\mathbf{C}^{(g)} + \frac{c_{\mathrm{cov}}}{\mu_{\mathrm{cov}}} \left(\mathbf{p}_{\mathrm{c}}^{(g+1)} \mathbf{p}_{\mathrm{c}}^{(g+1)^{\mathrm{T}}} + \delta(h_\sigma^{(g+1)})\mathbf{C}^{(g)}\right) \\ &\quad + c_{\mathrm{cov}} \left(1 - \frac{1}{\mu_{\mathrm{cov}}}\right) \sum_{i=1}^{\mu} w_i \, \mathrm{OP}\left(\frac{\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}}\right) \end{aligned} \tag{34}$$

[1] The optimum should presumably be within the cube $\mathbf{m}^{(0)} \pm 2\sigma^{(0)}(1, \ldots, 1)^{\mathrm{T}}$. If the optimum is expected to be in $[0, 1]^n$ (initial interval) we may choose the initial search point, $\mathbf{m}^{(0)}$, uniformly randomly in $[0, 1]^n$, and $\sigma^{(0)} = 0.5$. Different search intervals Δs_i for different variables can be reflected by a different initialization of $\mathbf{C}$, in that the diagonal elements of $\mathbf{C}^{(0)}$ obey $c_{ii}^{(0)} = (\Delta s_i)^2$.

The following sections deal with the implementation of this strategy within the MZ tool.

7.3. Defining the Stop Criterion for the Problem

As with any optimization algorithm, the CMA is highly dependent on a proper definition of a stop condition that terminates the main loop of the algorithm and returns a final solution to the problem. Two main issues make the formulation of this criterion a delicate matter: if the algorithm is terminated too quickly then the final solution will not be a “good enough” approximation to the minimum or maximum; on the other hand, allowing too many iterations hardly produces more accurate results at the cost of computational performance.

The standard version of the CMA uses three stopping criteria:

- Break the loop if the value of the objective function (fitness) of a solution is below or above a threshold set by the designer for finding minima and maxima, respectively.
- Break the loop is the maximum standard deviation amongst the covariance matrix is bigger than the minimum standard deviation in the same matrix times a safety factor.
- Break the loop if the number of iterations exceeds a fixed maximum.

Analyzing the three default stop criteria it may be concluded that the best (and simplest) way to define a custom definition for a new criterion is to determine what the minimum number of runs should be to achieve convergence. It is undesirable to set a “minimum LPC” as per demanded by the first default criterion, as this limits the quality of the solutions that may be found and bounds the algorithm to find a final solution that is not close to the global minimum. This is more or less equivalent to placing a constraint on the problem by bounding the possible LPC outcomes. The second default condition is derived from the mathematical properties of the covariance matrix and thus it is not advisable to change its definition, due to the fact that it is based on internal parameters and workings of the CMA and its apparent suitability to serve as a stop criterion.

Defining a minimum number of iterations based on the number of turbines may seem to be the best approach, yet there is one problem regarding this methodology: the number of parameters that have an effect of convergence is too large to be able to properly generate a function that uses all of them to predict what an optimum number of iterations should be. Therefore, the approach undertaken here is simply to provide some observations on the convergence of the algorithm that may serve as an indicator, in order to make a gross estimation of what the stop criterion should be.

The simulations performed to assess the convergence as a function of the number of iterations were designed based on variations on two parameters that may have an effect on convergence speed. These parameters are:

- Number of turbines: has a direct effect on the performance on the CMA since the number of variables to optimize (coordinates) is directly related to this parameter.
- Initial step size: this is an internal parameter of the CMA, which is recommended (36) to be set to 0.5. The step size is adaptive in the CMA algorithm, yet there may be an effect on convergence speed depending on the selection of its initial value.

The effect of these parameters on convergence speed shall now be analyzed in order to see if a correlation exists that may be used to generate some indications for a possible stopping condition. However, before a proper analysis is performed a minimum number of iterations must be defined such as it is possible to “see” the algorithm converging and thus, analyze the effect of altering the parameters.

7.3.1. Number of Iterations for evaluation of stopping criterion

In the original version of the CMA the minimum number of iterations to stop the algorithm is given by the following equation:

$$stop_{eval} = 1000 * N^2$$

Equation 24.

where N is the number of variables to be optimized. For the turbine siting problem it would be equal to two times the number of turbines in the wind farm. There is no information regarding whether this equation is based upon empirical data, calculated from the algorithm’s properties or simply an observed minimum plus a safety factor. It does show however that no less than 1000 iterations would be required until this particular stop condition is applied, which for simple optimization problems is acceptable but not for the turbine siting problem as defined here. The reason behind the inapplicability of this stop criterion is computational time.

To illustrate this problem the CMA was used to solve two different optimization problems: finding the minimum of the Rosenbrock function with 20 problem variables and minimizing the LPC of a wind farm with 10 turbines (which also has 20 problem variables). The CMA was timed for a fixed number of iterations; the results are shown in Table 16.

Table 16. Timing of CMA performance for 100 iterations over two different optimization problems.

CMA Time Performance on a Machine with Intel

Core i5-2430M CPU @ 2.4 GHz 100 Iterations	
Optimization Problem	**Time (s)**
Rosenbrock function (N=20)	0.38
LPC Minimization (N=20)	889.84

The time difference between performing 100 runs for the optimization of the Rosenbrock function is several orders of magnitude smaller than running the same number of iterations for the LPC minimization problem. The reason for such a difference is evident: a simple algebraic function such as the Rosenbrock function receives an input parameter ‘x’ and calculates and output ‘y’, which is the equivalent of an objective function. For the turbine siting problem, the calculation of the LPC is by no means a simple algebraic function: it involves several calculations and optimizations of internal design variables (the most time consuming being performing a wake analysis) which results in significantly longer objective function evaluation times.

A test batch of unrestricted optimization runs was performed in order to attempt to determine at which point the solution was considered “converging” to a final LPC value. The results of the convergence of this test batch (10 turbine optimization) are presented in Figure 45.

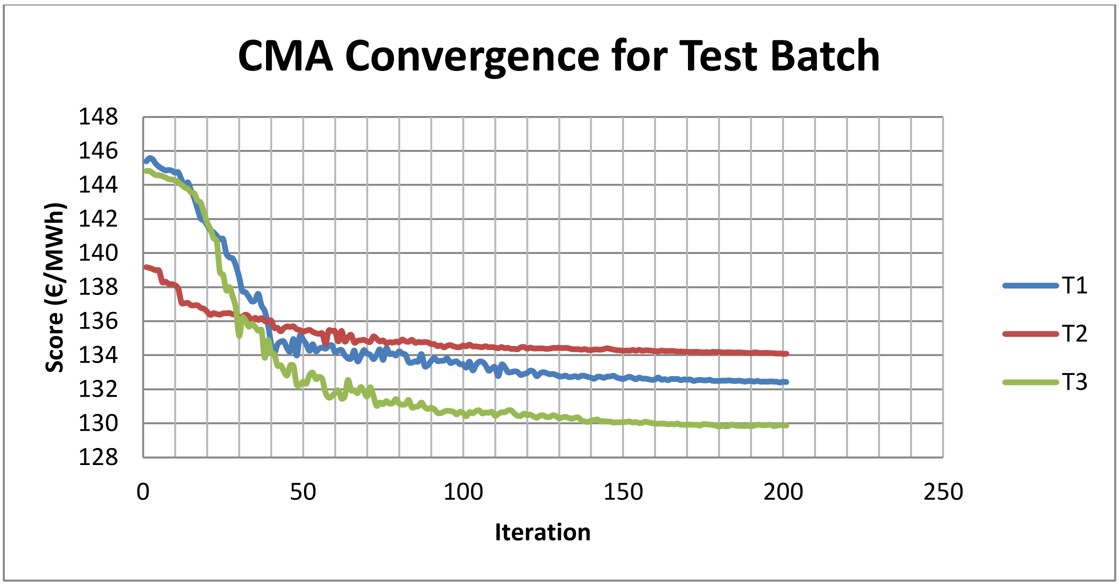


Figure 45. Score as a function of number of iterations for test batch, starting from an initial guess. Different series represent different simulations.

Notice that in this figure the number of iterations plotted has been limited to 200 since visual convergence of the simulations has been achieved before this number. Figure 46 presents a close up of Figure 45 for the last 50 iterations, in which the approximation of the simulations to asymptotic values is noticeable.

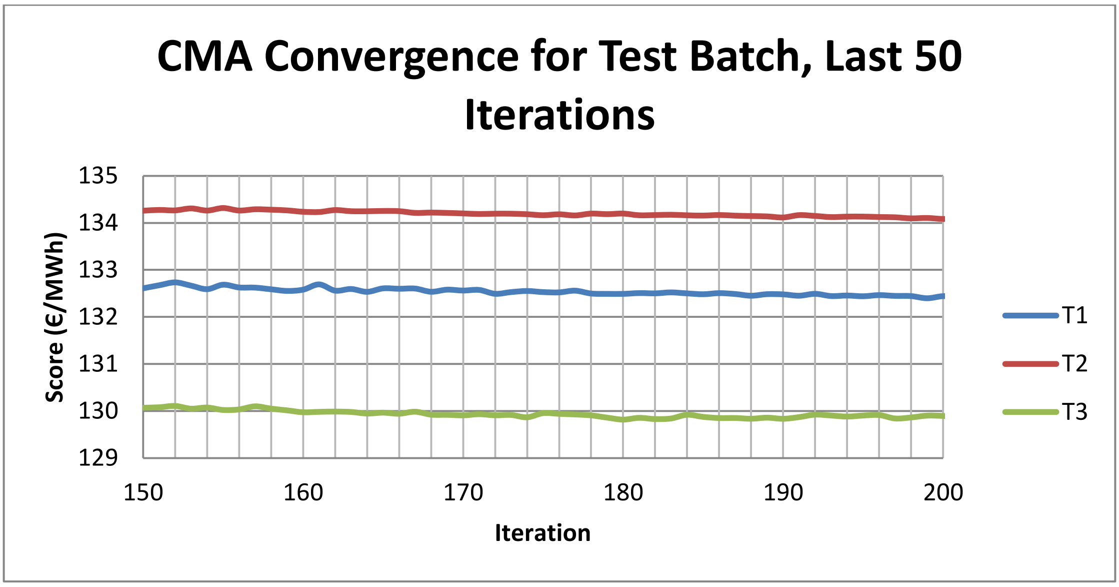


Figure 46. Close-up of Figure 45, for the last 50 iterations. Different series represent different simulations.

The net change in Score between iteration 150 and iteration 200 for all three test simulations is around 0.2 euro cents per kWh. This is a small change in LPC and therefore it may be assumed that convergence has been achieved.

If a more quantifiable condition of visual convergence is defined as the point in which the average percentage of improvement between five consecutive iterations is below 0.05% (five is used to ensure that premature convergence is not a stop criterion due to a “lucky” iteration) then at approximately 150 iterations it is possible to say that the LPC has converged, as shown in Figure 47 and Figure 48.

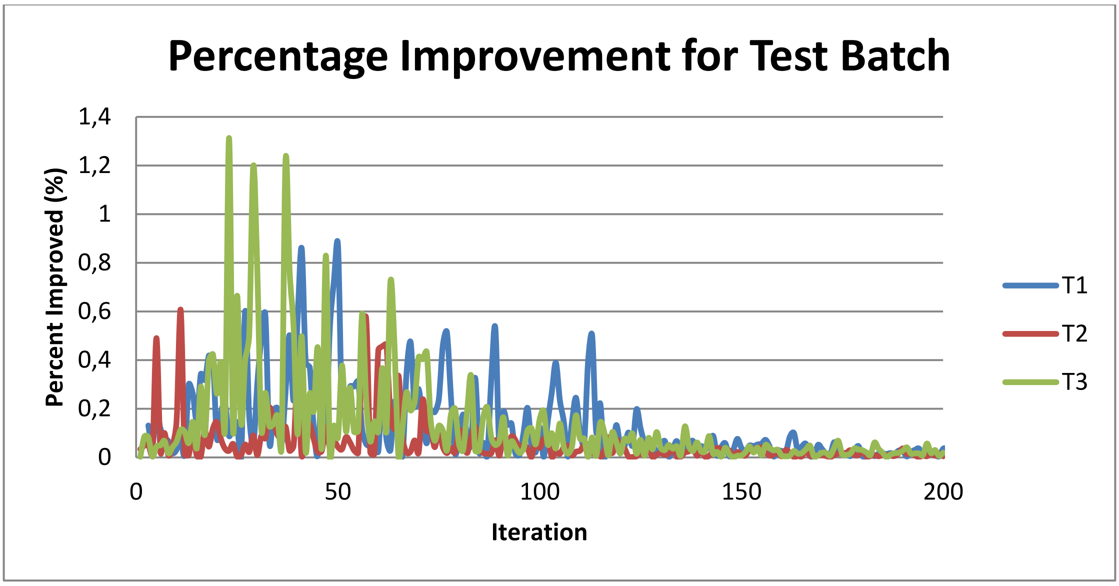


Figure 47. Percentage of improvement as a function of iteration number for the test batch. Different series represent different simulations.

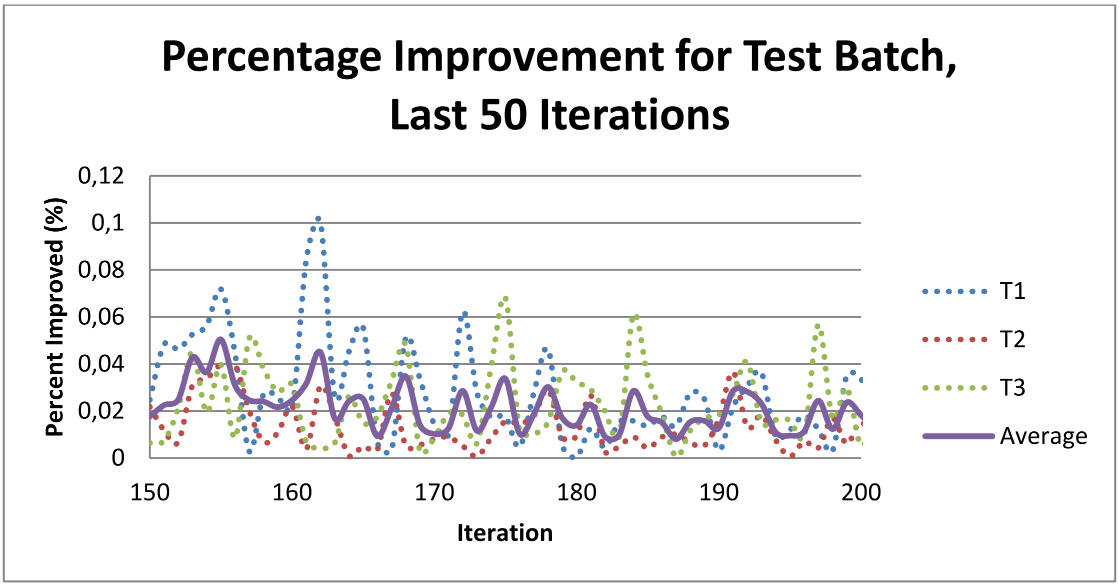


Figure 48. Close-up of Figure 47, fo the last 50 iterations. Additionally, the average of the three test simulations is plotted with the solid line.

The results suggest that 150 iterations are enough to visualize convergence of the simulation batches based on this condition.

7.3.2. Initial Step Size

A total of 12 simulations were performed while keeping the number of turbines and the number of sectors constant and only varying the initial step size used by the CMA. The definition of these simulations is shown in Table 17.

Table 17. Simulation configurations to assess the effect of initial step size on convergence.

Simulation Batch	Number of Simulations	Number of Turbines	Initial Step Size
A	3	10	0.5
B	3	10	0.3
C	3	10	1.0
D	3	10	1.5

The results of the simulations are presented below. Notice that simulation batch A is equivalent to the test batch defined in the preceding section (in which the step size was also 0.5).

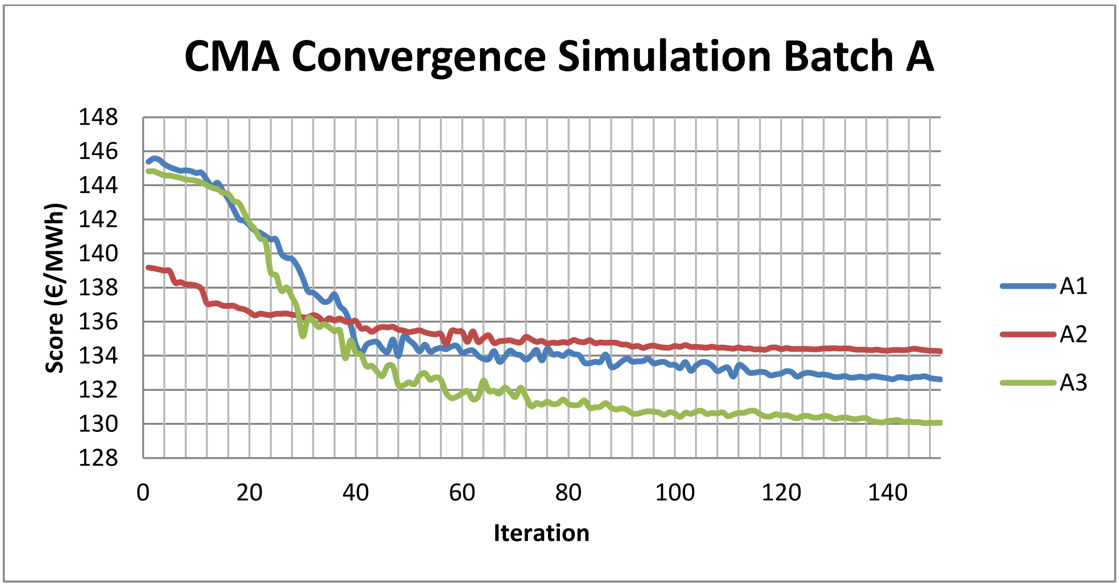


Figure 49. Score as a function of number of iterations simulation batch A, starting from an initial guess. Different series represent different simulations.

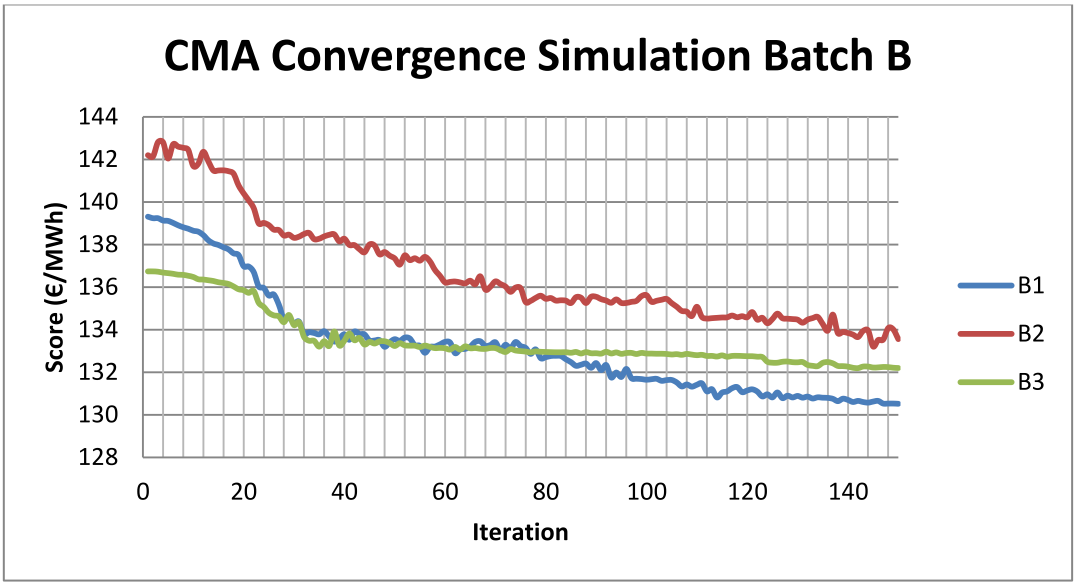


Figure 50. Score as a function of number of iterations simulation batch B, starting from an initial guess. Different series represent different simulations.

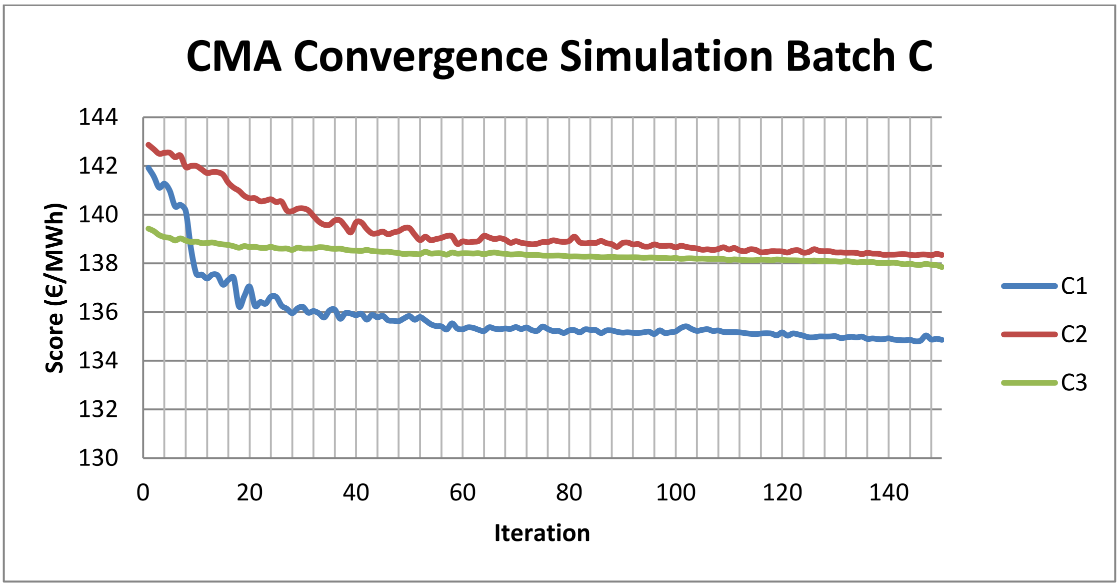


Figure 51. Score as a function of number of iterations simulation batch C, starting from an initial guess. Different series represent different simulations.

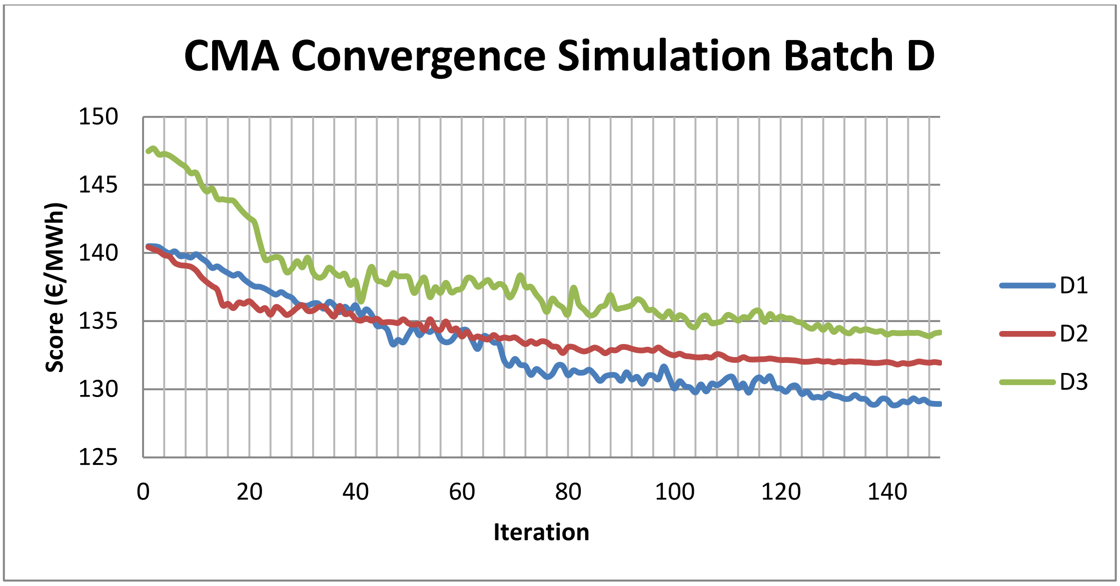


Figure 52. Score as a function of number of iterations simulation batch D, starting from an initial guess. Different series represent different simulations.

An analysis of the results reveals that there is no evident effect of the initial step size on the convergence speed of the algorithm. It must be stated however that convergence is being evaluated in a visual, somewhat qualitative way by simply assessing the number of iterations after which the score stops its phase of steep decline and starts becoming "flat", since the previous criteria for convergence is generally met towards the end of the 150 iterations defined. An approximate number of iterations after which this transition occurs for each set of simulations is presented in Table 18.

Table 18. Approximate range of iterations at which relative convergence begins to occur for simulation batches A, B, C and D.

Simulation Batch	Number of Iterations to Approximate Convergence
A	60-70
B	120 - 130
C	60 - 70
D	120 - 130

There is no clear correlation between the variables that may be used as evidence to conclude that the initial step size plays an important role in defining the optimal number of iterations. Nevertheless, simulation batch D seems to show that large initial step sizes result not only in delayed convergence but also in a decline phase with a smaller slope compared to the other simulations. It may then be concluded that the adaptive properties of the step size of the CMA quickly stabilize any changes made to the initial value of the parameter and the convergence speed achieved remains the same, therefore, the default step size of 0.5 will be fixed for the remaining simulations.

7.3.3. Number of Turbines

In an analogous manner to the analysis for the effect of initial step size, a set of simulation batches to test the effect of this parameter on convergence speed was defined as follows:

Table 19. Simulation configurations to assess the effect of number of turbines on convergence.

Simulation Batch	Number of Simulations	Number of Turbines
A	3	10
E	3	15
F	3	30
G	3	45

Notice that simulation batch A is once again used here, thus the results are not replicated here but can be observed in Figure 49.

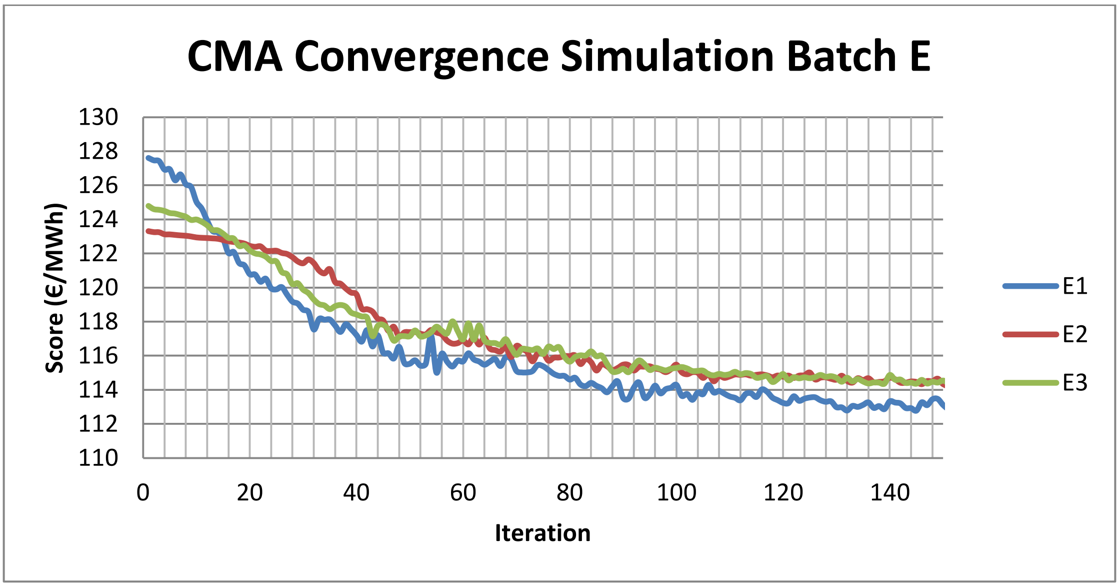


Figure 53. Score as a function of number of iterations simulation batch E, starting from an initial guess. Different series represent different simulations.

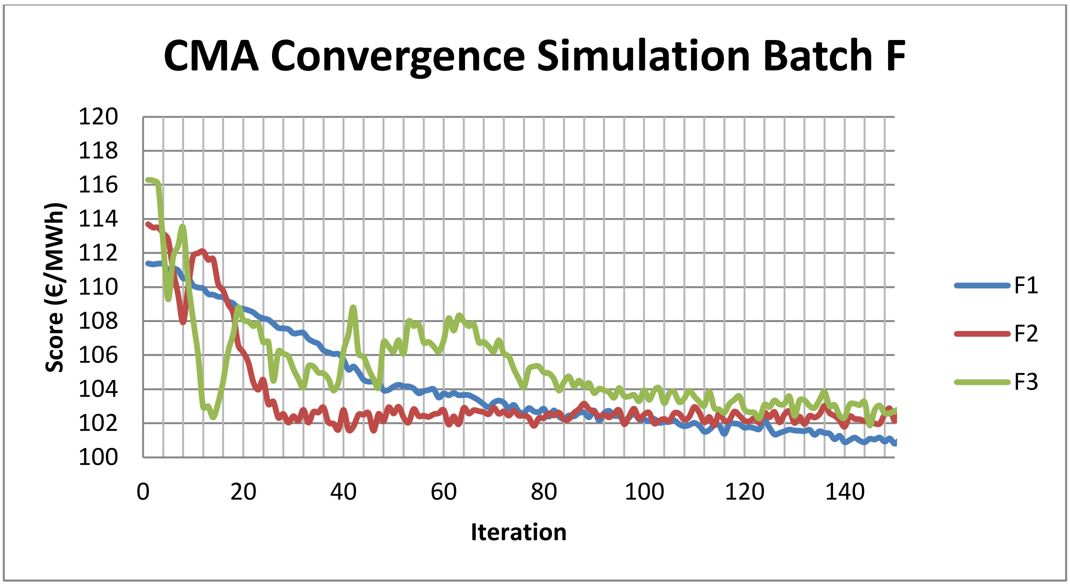


Figure 54. Score as a function of number of iterations simulation batch F, starting from an initial guess. Different series represent different simulations.

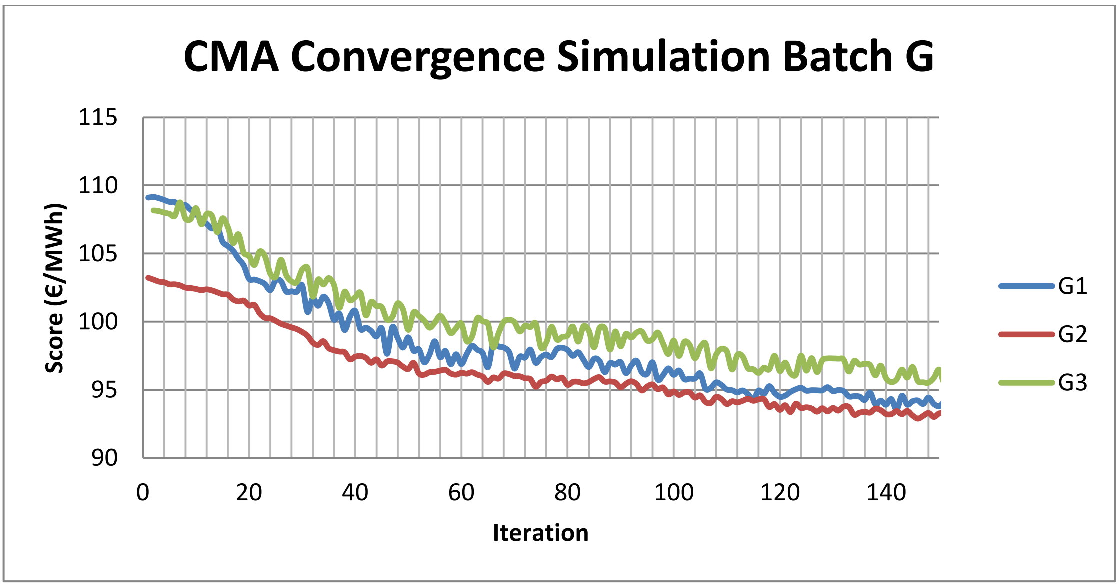


Figure 55. Score as a function of number of iterations simulation batch G, starting from an initial guess. Different series represent different simulations.

There are several noticeable effects of the number of turbines on the behavior of the CMA algorithm (most of which will be discussed on the following chapter). Focusing on algorithm convergence it appears that the transition between the quick decay phase and the flat line phase occurs at a higher number of iterations as the number of turbines increases. Notice, however, that there may be exceptions to this observation as shown by simulation F2 which appears to converge very fast and oscillate around a particular value of the score. Interestingly, there appears to be a correlation between convergence of the algorithm and number of turbines as demonstrated by the trend in Figure 20. It is worth pointing out that such a correlation is not fault proof and general, as demonstrated by the non-typical convergence behaviors of simulations A2, F2 and F3.

Table 20. Approximate range of iterations at which relative convergence begins to occur for simulation batches A, E, F and G.

Simulation Batch	Number of Iterations to Approximate Convergence
A	60-70
E	100-110
F	130-140
G	150>

7.3.4. Convergence for more than one Optimization Loop

All the previous tests were performed for 150 iterations within a single CMA loop. Nevertheless, once the CMA has produced a solution the whole program will enter another optimization global loop (as shown in the sequence of disciplinary optimization, Figure 7) after which a second CMA optimization will be performed and so on. Therefore it is worth analyzing what happens for multiple CMA runs with the convergence of the solution, since a possibility exists that changes in the cable design, support structure, tower height, or other design variables of the wind farm produce a change in the LPC.

A single simulation with test conditions identical to simulation batch A was performed, limiting the maximum number of iterations per CMA run to 150. Three global optimization loops were allowed. The result of such a simulation is presented in Figure 56.

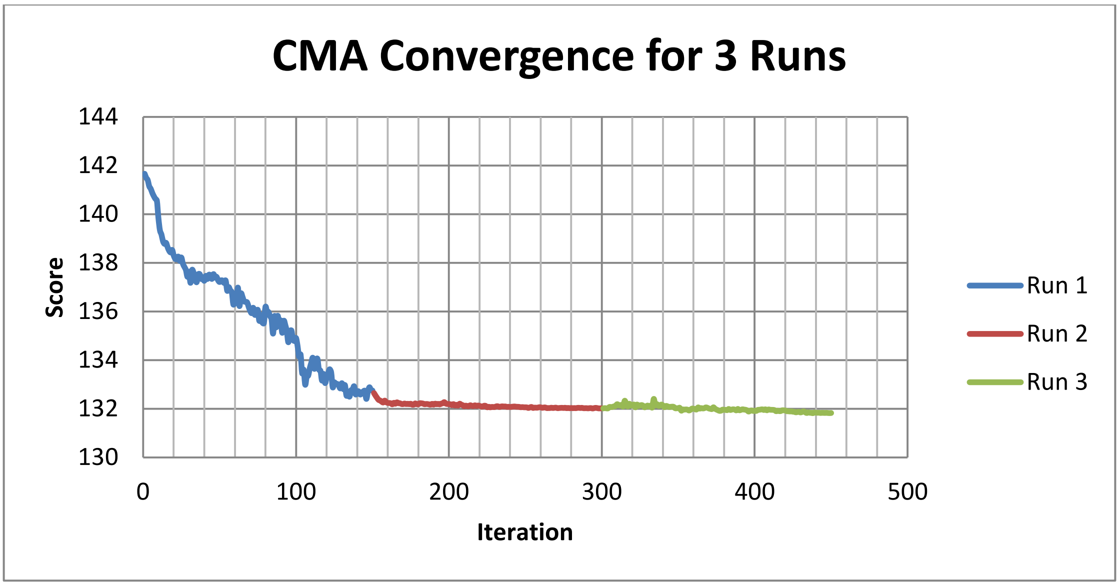


Figure 56. Score as a function of iteration number for three CMA runs, using parameters of simulation batch A.

The figure illustrates that two CMA runs are sufficient to achieve a completely convergent LPC, with the partial convergence criteria being met at around 150 iterations as predicted previously.

7.3.5. Recommendations for Criteria Definition

Based on the simulations and results presented in sections 7.3.1 through 7.3.3 the following set of recommendations are presented if in future work it would be desired to include a stop condition for the CMA related to the parameters analyzed:

- The initial step size has no discernible effect on convergence speed and thus it is recommended to set it to a value of 0.5, as suggested by literature.
- The number of turbines in the wind farm should be the main driver in the selection of a stop criterion (not only backed up by the existing criterion of Equation 24, but also by the simulation results and observations). A correlation seems to exist between the number of turbines and the number of iterations until convergence (visually

assessed); yet in order to extract an equation that would model such a behavior requires more data and tests that are not developed within the scope of this project.

Thus, the recommendation is to explore the possibility of defining a stopping condition that is a function of the number of turbines that yields a number of iterations high enough to achieve convergence under any scenario. The simulations performed with relatively small numbers of turbines show that at least 150 iterations are required for most test cases to guarantee convergence as defined previously (less than 0.05% variation for a number of sequential iterations).

7.4. The Area Constraint

In order to obtain a feasible final solution it is required that most of the offspring that are recombined in the CMA are feasible solutions themselves. If the recombination is based only on fitness, it is highly likely that some solutions which have turbines placed in invalid spots (for example, outside the wind farm area) yield lower LPC values due to the reduced wake effects and thus are chosen by the optimization algorithm as strong candidates for recombination.

There are several ways to implement a constraint in an optimization algorithm; the method chosen here is one of the more simple and straightforward, known as the static penalty function approach. Consider once again the general formulation for a minimization problem given by Equation 1. The concept behind applying a penalty function is to generate a modified version of the objective function (denominated the penalized objective function), given by the following expression (66):

$$f_p(x) = f(x) + \sum_{i=1}^{m} C_i \delta_i$$

Equation 25.

Where *$f_p(x)$* is the penalized objective function, *$f(x)$* is the objective function, *m* is the number of constraints, *C_i* is a constant imposed for the violation of constraint *i* and *δ_i* is a parameter equal to 0 if the constraint is respected, or 1 if the constraint is violated. Currently only one constraint will be applied to the optimization, therefore Equation 25 is reduced to Equation 26.

$$f_p(x) = f(x) + C\delta$$

Equation 26.

The issue is now the determination of the constant that must be applied as a penalty. It is evident that under-penalizing the LPC must be avoided since under certain cases this will allow certain highly undesirable solutions to recombine. However, over-penalizing the objective function can also lead to undesirable results. If a solution that has been over-penalized is still selected by the algorithm for recombination (which could happen if its high "unfeasibility" gave it a much lower LPC than the pool average) then stability problems of the algorithm may occur due to the highly perturbed solution that was used as a parent.

The approach chosen here to determine the magnitude of the penalty is as follows:

- Calculate the standard deviation amongst the LPC's of simulation batches A, E, F and G. The reason these batches are used is because the magnitude and range of the score (and therefore the LPC) is only a function of the number of turbines amongst the three parameters used to analyze convergence in the previous section.
- Average the standard deviations per simulation batch.
- Generate a plot of average standard deviation versus number of turbines to see if there is any correlation. If not, select a penalty value that is higher than the maximum standard deviation in the plot.

The before mentioned plot is presented in Figure 57.

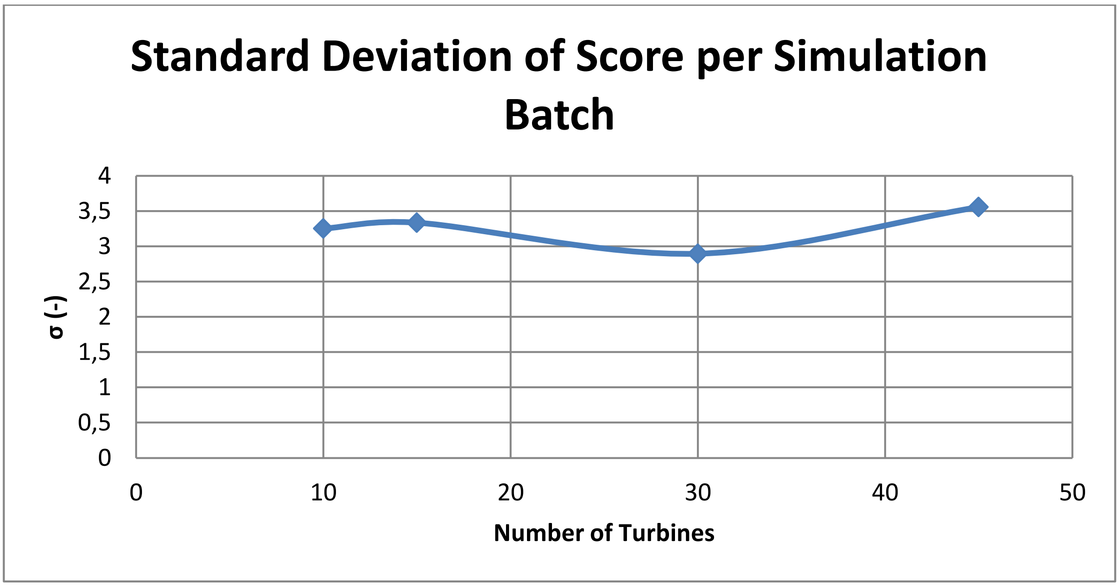


Figure 57. Average standard deviation for score values amongst simulation batches A (10 turbines), E (14 turbines), F (28 turbines) and G (40 turbines).

There is no clear trend that can be adjusted to this small dataset, although it appears that the variation in the score remains bounded in a relatively small bandwidth regardless of the number of turbines used. Therefore, applying a static penalty function will likely guarantee a feasible final solution for the turbine siting problem.

The penalty constant is here determined by first obtaining the highest standard deviation identified from the test runs. Afterwards, floor rounding is applied to improve the stability of the solutions, which comes at the cost of granting more non-feasible offspring the chance to recombine. The logic behind it is as follows: applying a smaller penalty will inevitably lead to unfeasible solutions displaying lower LPC’s than the feasible offspring in the same generation to improve their chances of recombining, but if the constant were ceiling rounded the result would be a greater difference between penalized and un-penalized LPC’s that would result in a higher degree of convergence instability. The former is preferred to the latter since some of the unfeasible turbine positions can be ultimately corrected manually by the designer.

Following the procedure described previously, the penalty constant will be set as:

$$C = 3.63 \approx 3$$

8. Discussion on Performance

8.1. Introduction

The previous chapter discussed the implementation of the Covariance Matrix Adaptation Evolutionary Strategy, which is a heuristic algorithm developed to solve multivariate optimization problems. Several observations and conclusions on convergence of the LPC were analyzed in order to try and determine an appropriate stopping condition for the algorithm.

This chapter discusses the suitability of using the POS1 and CMA to solve the turbine siting problem by analyzing the solutions generated by the algorithm. Remarks on computational performance will also be briefly explored here, although this is not the main focus of the discussion. The main issues treated in this chapter include computational performance in terms of speed, observations on the LPC's of the designed solutions and analysis of the wind farm layouts themselves.

8.2. Algorithm Speed

In section 7.3.1 it was briefly discussed that the original stop criterion for the CMA could not be kept due to the time intensive nature of the objective function evaluation, with the turbine siting problem showing a run time several orders of magnitude larger than the optimization of a simpler function.

Profiling of the entire optimization algorithm revealed that the bottleneck of the optimization is the objective function evaluation; in particular the wake analysis. González (42) pointed out that for the particular wake analyst implemented in the MZ tool the order of the algorithm had an $O(n^2)$ dependency on the number of wind turbines, due to the fact that several loops over the list of turbines in the wind farm are required. This dependency is presented in Figure 58, in which two different wake analysis algorithms are compared: the "base algorithm" is an initial version developed in (42) while "second algorithm" is the algorithm currently used by the MZ tool. The parameter *O1/O2* is the ratio between the base and the second algorithm order of computation. Notice that the order of the calculation is proportional to computational time, yet it is not a measure of time in itself.

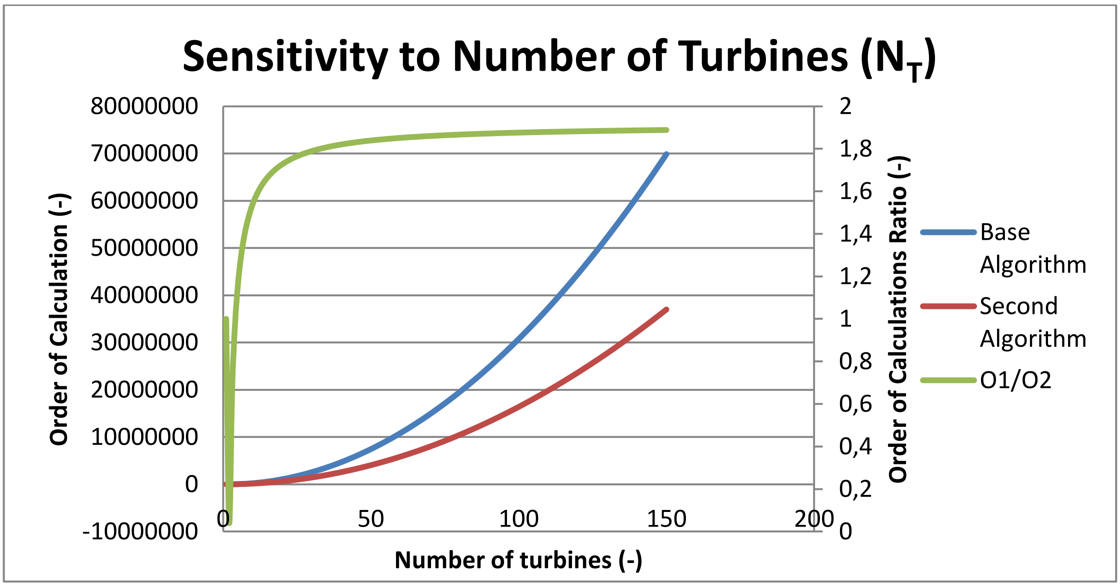


Figure 58. Sensitivity analysis for the wake analyst currently implemented in the MZ tool; dependency on number of turbines. Reproduced from (42).

In order to determine if the entire optimization routine's time performance is in fact restricted by the wake analyst, four different simulations timed where the number of wind turbines was varied. The timed simulations were A3, E1, F1 and G1 (corresponding to 10, 15, 30 and 45 turbines, respectively). The resulting times were plotted and are shown in Figure 59.

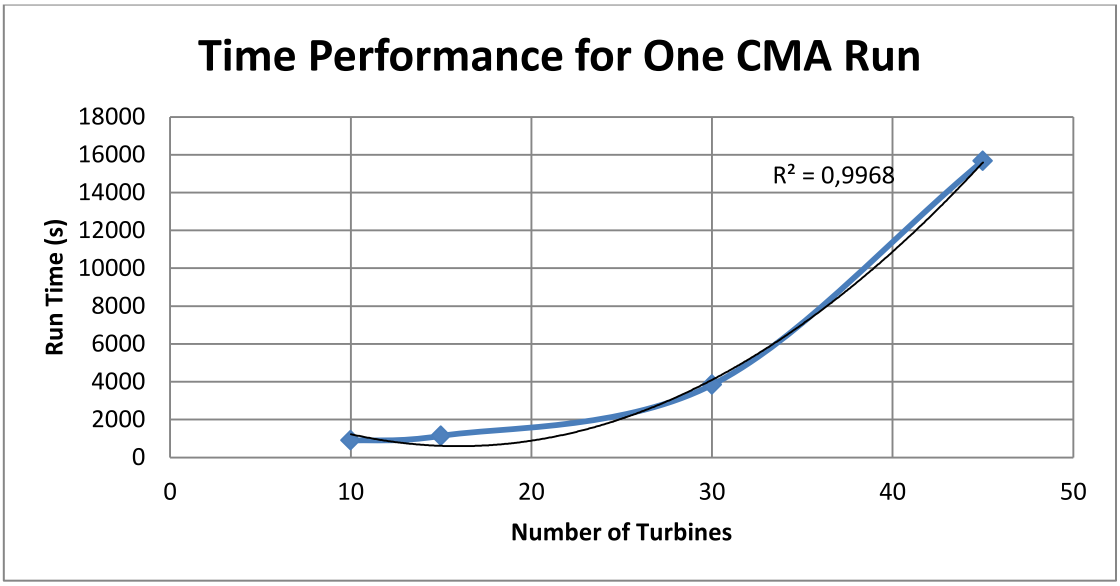


Figure 59. Time to complete a CMA run (limited to 150 iterations) as a function of the number of turbines in the wind farm, on an Intel Core i5-2430M @2.40GHZ machine.

The plot illustrates that the curve fit between run time and number of turbines is, in fact, quadratic. This result is consistent with the expected order of the algorithm calculated by González (42). It appears that indeed the wake analyst is the bottleneck of the optimization routine, showing a time complexity response entirely dominated by its predicted behavior.

Nevertheless, it is possible that another part of the CMA execution also displays a O(N^2) time complexity that could potentially be a bottleneck as well, yet tests using different objective functions (like Rosenbrock) don't seem to point out that this is necessarily the case.

The current version of the wake analyst coupled to the existing optimization routine will lead to a poor computational performance for medium to large wind farms; according to the trend line a wind farm the size of Horns Rev would take about 20.4 hours to optimize (notice that this is for a single CMA run). Therefore, although the "pure" CMA displays a time-efficient behavior (demonstrated by evaluation of an algebraic function), the objective function evaluation and in particular the wake analyst require further optimization and speed-up strategies in order to enable the tool to optimize large wind farms within practical time scopes.

8.3. Final Solution LPC's

The simulation batches presented in section 7.3 evidence a particular, unexpected characteristic of the CMA algorithm applied to the turbine siting problem: dependency on the initial guess. It is not uncommon for heuristic algorithms to yield different final solutions for the same problem depending on the initial values assigned to the problem variables; this is due mainly to their non-exact nature (introduced by their "randomness" approach to finding a solution) and the existence of multiple local minima or maxima in the solution space.

In the simulation batches used to evaluate convergence of the CMA after 150 iterations a relative level of convergence was achieved for the score parameter of the solutions, with many of them displaying a trend to oscillate around an asymptotic value with minimal changes in the LPC value. However, for simulations within the same batch it is impossible to determine a single score value to which all simulations converge. Take for example simulation batch A (Figure 49): the LPC of the final solutions of each simulation are approximately 132.5 for A1, 134.3 for A2 and 130.2 for A2. The net differences between them (around 0.2 € cents), represents a non-uniform behavior of the final solution that the CMA yields; the magnitude of the net change may seem small but for energy pricing issues a change in tenths of a cent is still significant. Furthermore, the final wind farm design also differs for every simulation, indicating that the algorithm does have a strong dependency on the initial guess. This hypothesis may hold true for "short" numbers of iterations only; however, increasing an order of magnitude in the maximum number of iterations per CMA run is currently highly unpractical in terms of computational time as described previously, and there is no evidence to believe that eventually all simulations will converge to one unified solution considering that asymptotic trends start to emerge after a small number of runs.

Some heuristic algorithms that show dependency on the initial guess circumvent the problem by what is called a multi-start approach: instead of using a single feasible solution as a starting point, several of them are generated and then either used as an initial pool for recombination or the best one is selected as the original parent. Applying a multi-start approach to the CMA strategy may seem tempting, yet there is no guarantee that it will lead to the global optimum and a single solution for every optimization due to the fact that often optimizations that start with a “bad” initial guess end up being the best of a batch, as demonstrated by simulations A3, C1 and E1. This strategy could serve however as a way of finding the best solution among non-optimal solutions (considering the scenario in which several optimizations failed to find the global optimum), which is indeed more desirable.

What may be concluded from this discussion is that the problem in question seems to have several local minima and possibly more than one optimal solution, which is not strange considering that the large number of parameters and variables involved in its definition generate a complicated, non-monotonous solution space.

8.4. Resulting Wind Farm Layouts

The previous sections have dealt with issues and observations related to the intrinsic CMA performance; however, so far there hasn´t been any discussion regarding how the designed layouts look like. This issue is tackled here. For the layouts presented in this section no constraint was imposed on the minimum turbine separation.

A forty turbine wind farm design subjected to area constraints in the form of a polygon defining the limits of the project is presented in Figure 60. Two data series are plotted (for this figure and all subsequent layouts presented in this chapter): the initial guess used by the CMA as a starting point and the optimized layout. For the layout plot presented in Figure 62, a visual queue indicating the displacement of each turbine from its initial to its optimized position will be provided when it is considered that the magnitude of such a displacement makes it difficult to determine the path followed by the turbine. The other plots don’t require such a visual queue since the initial and final position of a single turbine are often quite close and additional lines only convolute the plot.

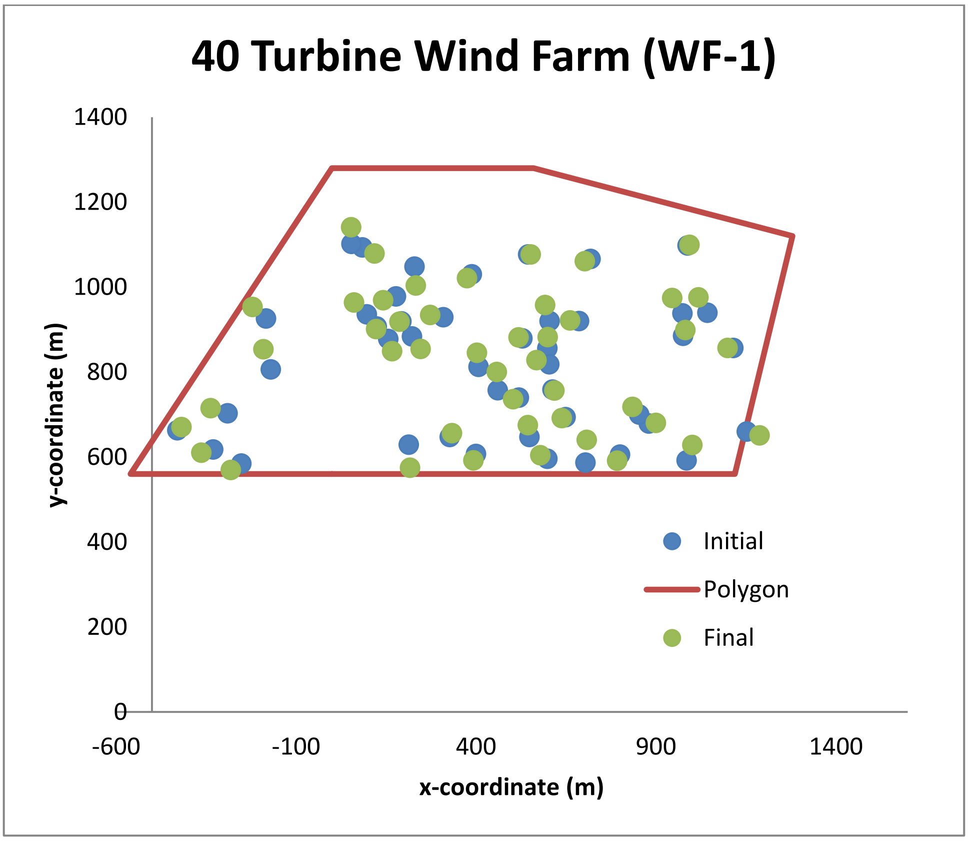


Figure 60. Initial guess and final solution after one CMA run for a 40 turbine wind farm subjected to area constraints (red polygon). Initial guess is plotted in blue, final solution in green.

Several observations may be drawn from the layout presented in Figure 60:

- Both the initial guess and the final solution display a uniform spread of the turbines throughout the area, although a concentration of points is perceived in the central area of the polygon. This is due to the fact that the initial guess samples from a normal distribution a value between 0 and 1 to determine the x and y coordinates of every turbine.
- The overall displacement of the turbines from their initial to their final position varies greatly depending on the turbine itself: some of them are barely moved (or less than 10 meters), while others are displaced more than 100 meters. This behavior is in line with a previous implementation of the CMA for the turbine siting problem by Rodrigues et.al (36), and was completely expected.
- One of the turbines in the layout displays an unfeasible position, product of an unfeasible offspring recombining at some point and exerting its influence in the final solution; the turbine in question is the easternmost one in the wind farm. The initial guess for this turbine is only two meters away from the boundary (inside the polygon). This particular turbine is an example of the effect of lowering the penalization on the

LPC to achieve stability, yet it is easy to see that a manual correction is very straightforward to achieve feasibility once more.

- Although a subjective observation, it seems that the turbines are being "pushed away" from each other (and in most cases towards the edges of the polygon). This is, for this case, a result of the turbines distancing themselves from their neighbors in order to reduce their wake losses and thus increase energy yield, thereby minimizing the wind farm LPC. However, depending on the initial separation distance the contrary is expected to happen: turbines will move closer together to lower cable costs.

8.4.1. Complex Boundaries

To further test the capabilities of the CMA in producing layouts several more complicated polygons were applied to bound the available wind farm area. The number of wind turbines was also varied for different simulations. These "more complicated" available areas and the designed layouts are presented in Figure 61, Figure 62 and Figure 64.

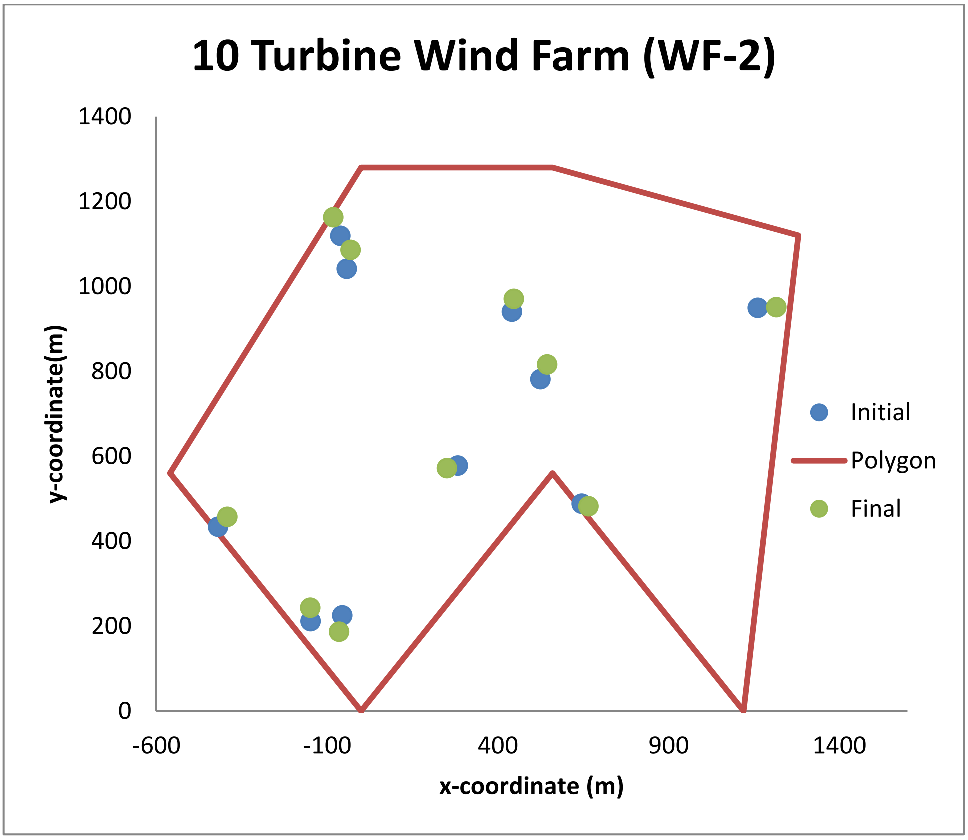


Figure 61. Initial guess and final solution after one CMA run for a 10 turbine wind farm subjected to area constraints (red polygon). Initial guess is plotted in blue, final solution in green.

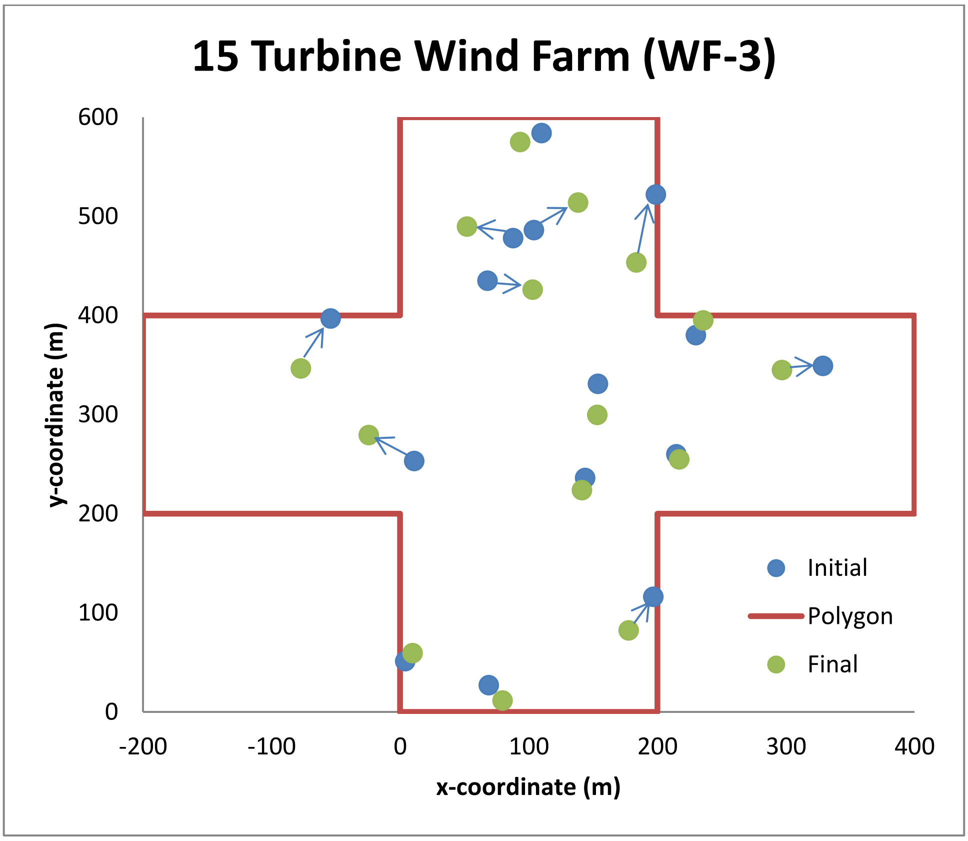


Figure 62. Initial guess and final solution after one CMA run for a 15 turbine wind farm subjected to area constraints (red polygon). Initial guess is plotted in blue, final solution in green.

In these more complicated polygon shapes the observations derived from the large wind farm presented in Figure 60 still hold valid, thereby proving that the CMA strategy implemented produces feasible layouts that respect the area constraints.

8.4.2. Turbine positions for more than one CMA Run

The previously presented layouts were generated for one CMA run using 150 iterations; however, it is also interesting to see what happens to the designed layout after more than one CMA run. To assess this effect the simulation that generated WF-2 (Figure 61) was allowed two perform two more passes over the main optimization loop. The results of this sequential CMA optimization are shown in Figure 63.

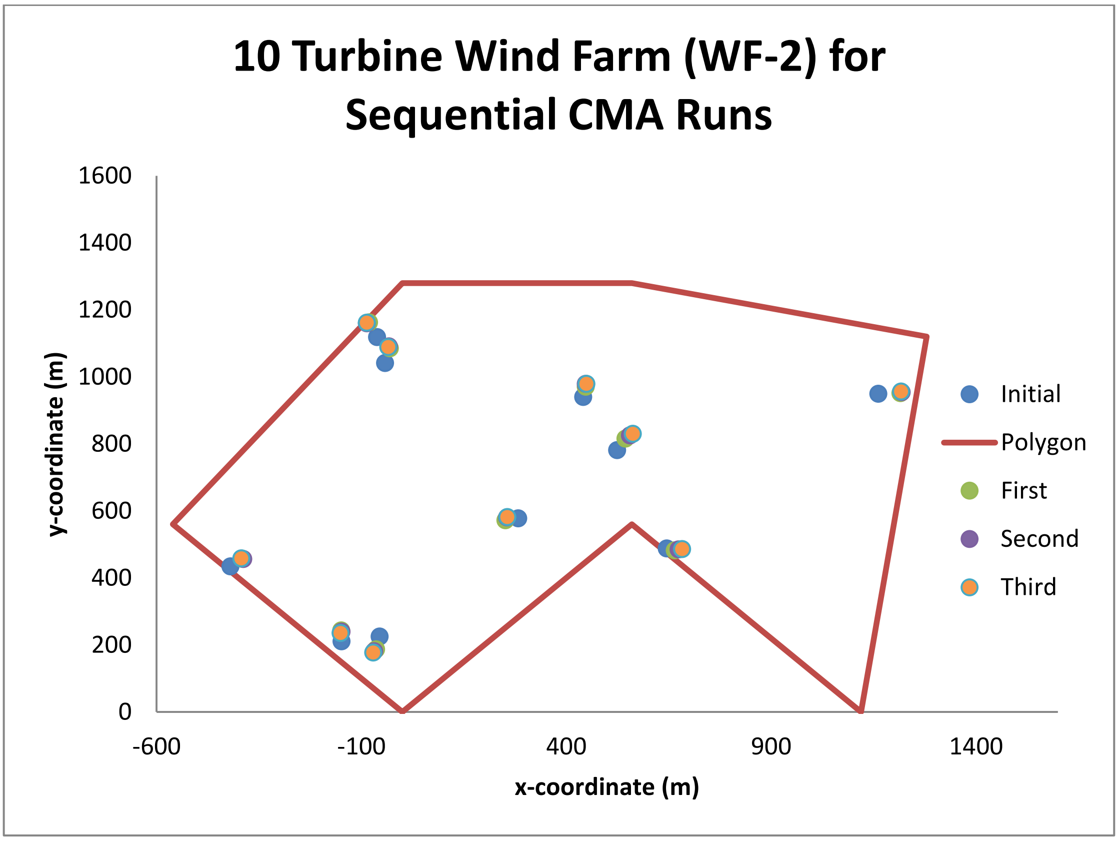


Figure 63. Initial guess (blue) and solutions after three sequential CMA runs (first in green, second in purple, third in orange) for a 10 turbine wind farm (WF-2).

The results indicate a behavior that is consistent with the observed LPC convergence after multiple runs for a single simulation presented in Figure 56. The rapid LPC decline phase is visualized by a notorious change in the turbine positions between the initial guess and the result of the first CMA run; for all subsequent runs the change in the turbine positions is quite small, with some of the turbines moving in the order of less than 5 meters between optimizations.

8.4.3. Wind farm layouts for unchanged conditions

In section 8.3 the dependency of final solutions on initial turbine positions was discussed, which served as an explanation for the different LPC values obtained when all design parameters and variables remain unchanged. In order to gain more insight on this issue three simulations with different starting points were performed to assess whether or not the turbine positions themselves tend to converge or not (unlike the LPC). The initial guesses for the three simulations are presented in Figure 64, while the final layouts after one CMA run are shown in Figure 65.

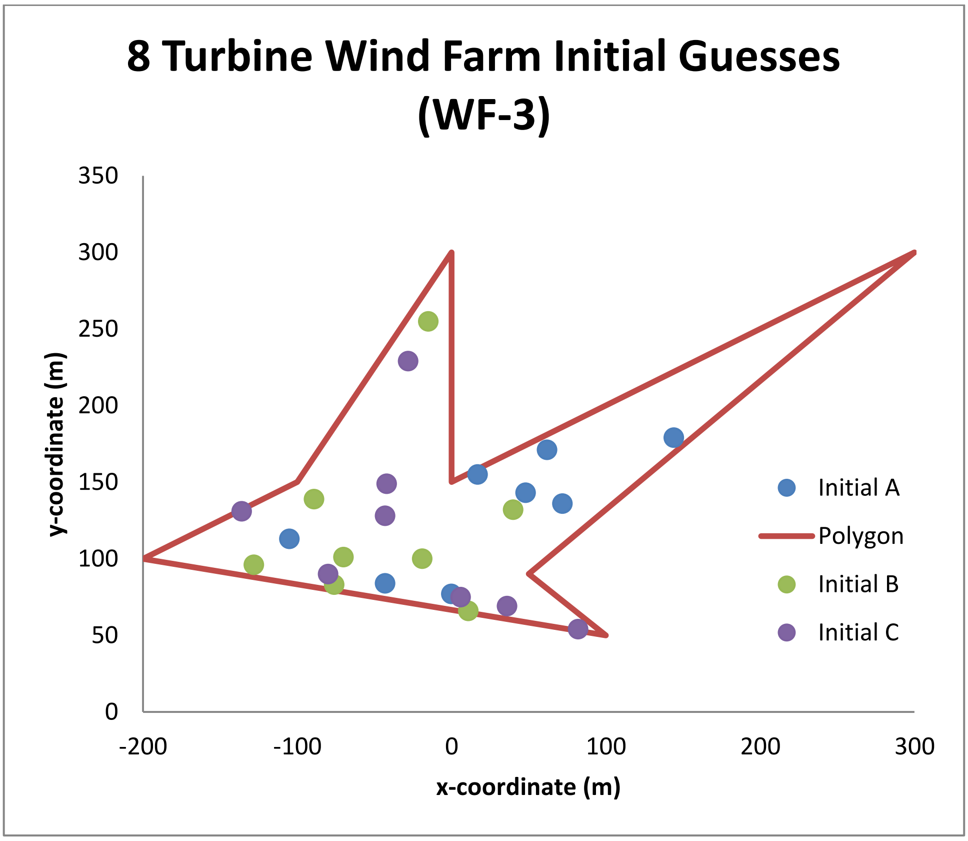


Figure 64. Initial guess for three simulations of 8 turbine wind farms, represented by different colored data sets.

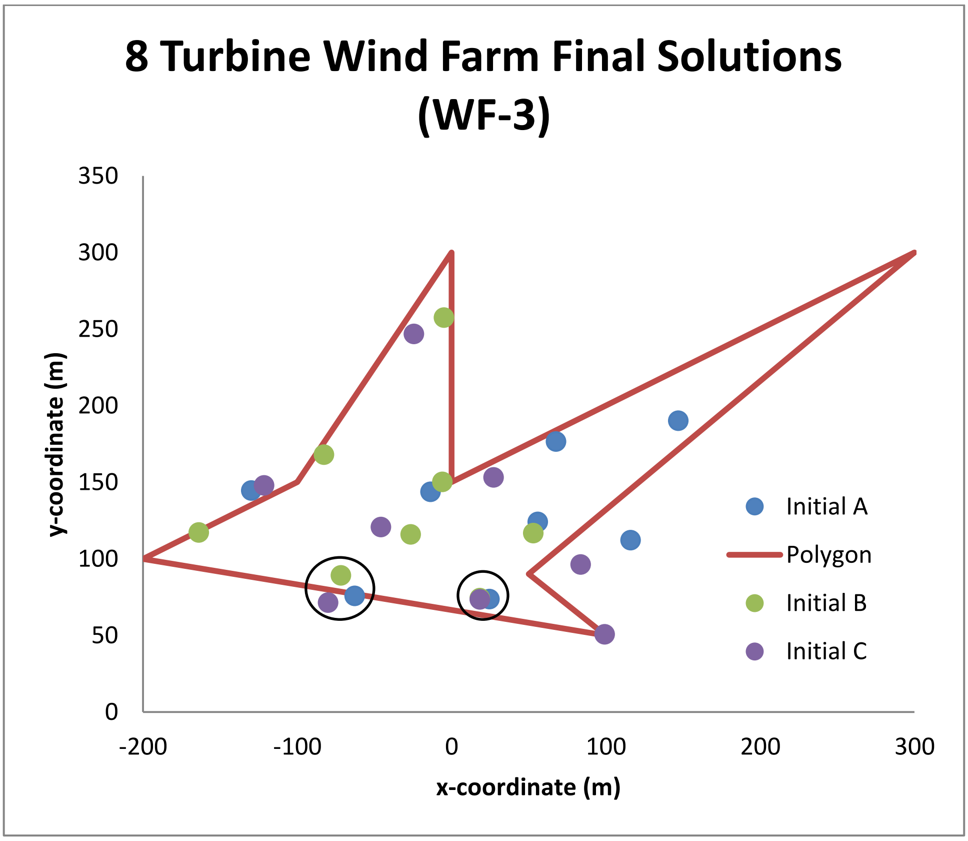


Figure 65. Final solutions after one CMA run for three simulations of 8 turbine wind farms, represented by different colored data sets. Each data series corresponds to its counterpart in Figure 64.

Two important conclusions may be drawn from these simulations:

- The polygon defining the wind farm boundaries is the most complicated one used so far, considering its sharp vertices and very “odd” shape to contain a wind farm. For this polygon the number of non-feasible points in the final solutions increased compared to previous simulations, indicating that it is possible that a complex shape leads to more unfeasible solutions being generated, and therefore, recombined.
- Interestingly, there appears to be a certain degree of clustering of turbines for different simulations, with some particular areas in the polygon displaying the convergence of three turbines (one for each simulation) with only a few meters of difference. These instances are highlighted by the black circles enclosing the turbines in Figure 65. This might be either an indication of the existence of clear minima around the contour of the solution space in these areas or an effect of the position of the OHVS, which is an important driver in the layout organization.

8.4.4. Cable Layout Design

One more aspect that is worth visualizing in the resulting wind farms is the cable layout design, which is generated by the POS1 algorithm described in chapter 6. Two cable layouts are presented in Figure 66 and Figure 67, which correspond to WF-2 and WF-3, respectively. For both cable layout designs the capacity per branch was fixed to a maximum of three turbines.

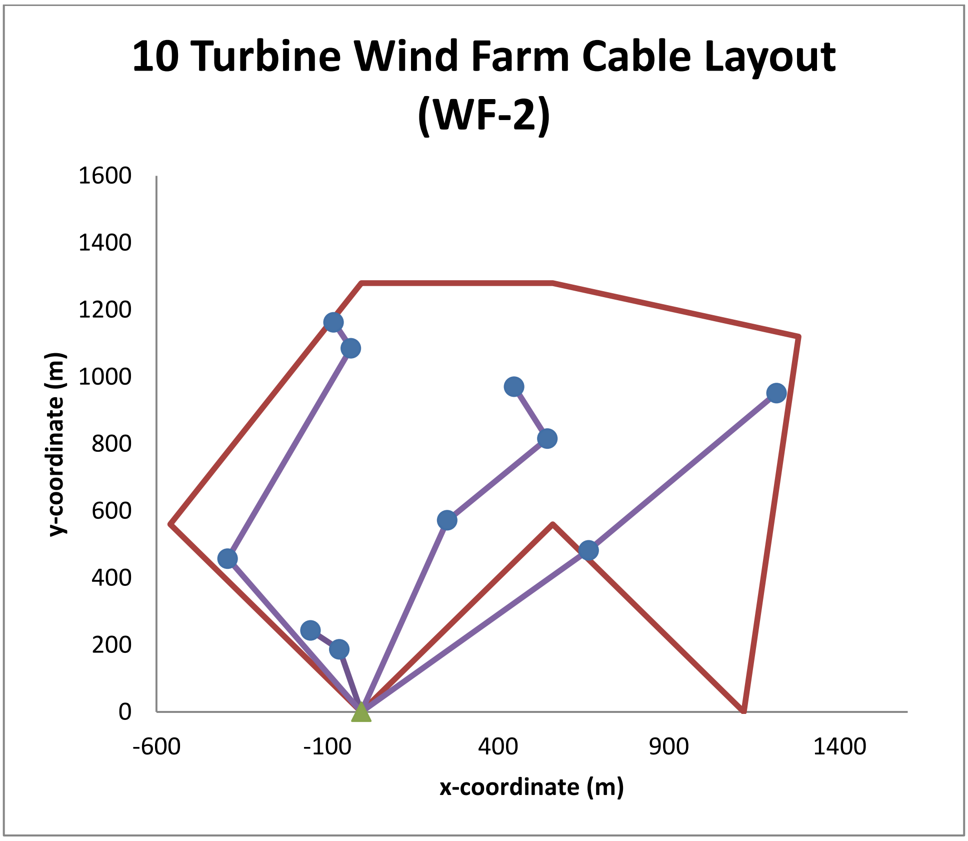


Figure 66. Designed cable layout for the 10 turbine WF-2; cables are shown in purple.

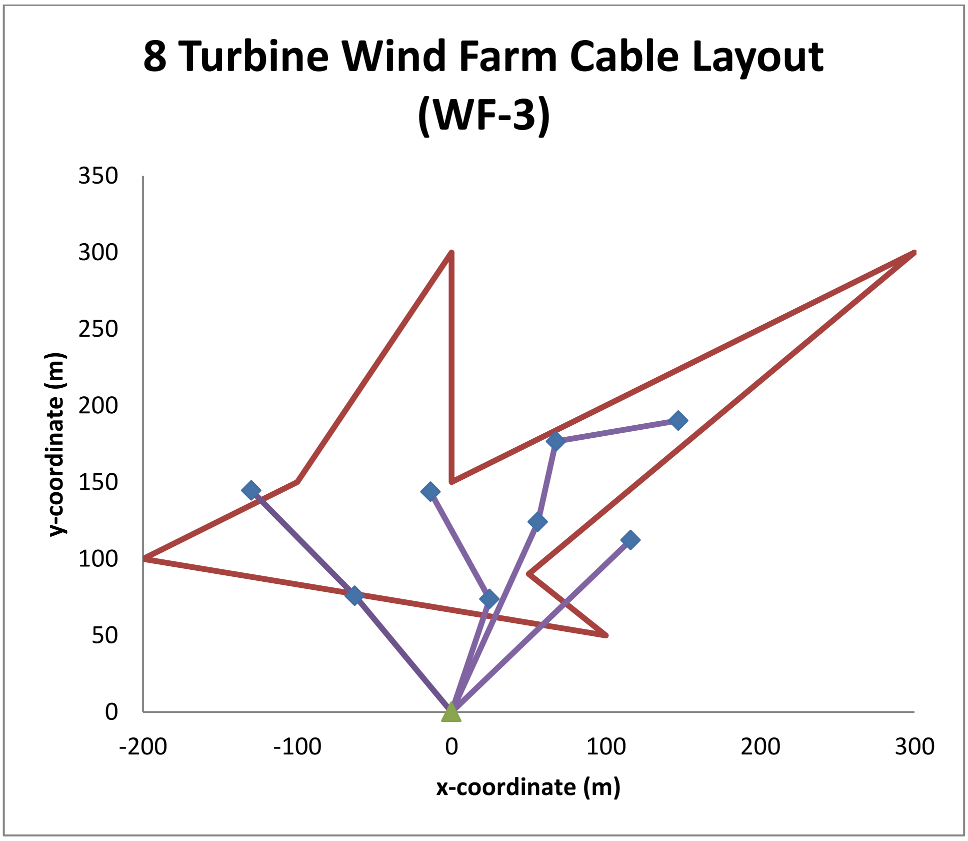


Figure 67: designed cable layout for one of the 8 turbine WF-3 versions. Cable is shown in purple.

The cabling layout presented in both figures is consistent with previous testing results for the POS1; the maximum capacity per branch is respected and there is no cable overlap. Furthermore, the layout is “logical” in the sense that it seems to trace the shortest possible routes between turbines. One issue with the cable optimization algorithm that was not treated as part of this project is notorious in both layouts: cable routes do not respect the area constraints. The cable that overlaps with the wind farm polygon is evidence of the problem to be addressed; it is easy to imagine that in order to treat this problem a modification on the POS1 is required that would allow it to connect two turbines using non-straight lines, that would allow the cable to “go around” the polygon boundaries.

In general, the CMA strategy and the POS1 algorithm developed in the scope of this project do an excellent job in yielding optimized layouts that respect constraints that are often placed on real wind farms.

9. Conclusions and Recommendations

9.1. Conclusions

This project documents the development of several modifications on an existing offshore wind farm design and verification computational tool in order to extend its capabilities, enabling it to generate more realistic layouts subjected to common constraints. A sequential problem solving approach was used, which ultimately proved to be an adequate method to solve the engineering problem at hand producing a solution that not only satisfies the initial objectives and requirements, but was also delivered in a timely manner.

A very extensive assessment of wind project developer' needs was performed using five different methodologies, from which a final list of functional requirements was extracted that served as the main driver for the specific project implementation goals. These were further prioritized based on a multi-criteria analysis that revealed that the most urgent aspects to be treated within the MZ tool were the ability to generate optimal wind farm layouts subjected to area restrictions and other real-world constraints.

The methodologies devised and used to obtain a list of user requirements was very comprehensive, since it allowed for the identification of not only software-specific requirements and opportunities for implementation but also limitations on current models that are used in the wind farm design process. The extrapolation from objective functions method was deemed to be the most useful, since it revealed all the "building blocks" required to arrive to the final result. Interestingly, while most objective function evaluations require a common set of minimum parameters there are some which are function-specific that could have not been identified if several objective functions were not assessed.

The overall methodology to identify user requirements for the wind farm optimization problem designed here is recommendable for future projects in the field due to the plurality of individual methods used and its ability to reveal both quantitative and qualitative requirements. Furthermore, this methodology could be adapted for use in other engineering problems that require their translation into a software tool, with the obvious observation that "observation from existing layouts" should be reinterpreted as observations from the particular problem at hand.

The process of prioritizing the requirements was designed to be as objective as possible, however, as with any other multi-criteria analysis there are always issues with biasing and subjective concerns reshaping the result and conclusions. This is a fact recognized by the author; however, the priorities identified within the scope of this project are in line with current research and literature on the topic, serving as evidence that while the initial method is not perfect there is some background to justify its results.

Two principal algorithms (including several subroutines each) were implemented as part of the scope of this project. They were developed sequentially, due to the fact that the pre-existence of an automated cable layout designer for any wind farm topology was required in order to develop an algorithm that would reposition the turbines themselves.

The cable layout design algorithm is an implementation of the POS1 designed and developed by Bauer et. al (63). This algorithm provides a solution to the Open Vehicle Routing Problem (OVRP) using a heuristic approach adapted exclusively for its use in wind farm optimization problems. The implemented version achieved an excellent computational performance even for high numbers of turbines, despite the fact of having a nested loop structure and a sequential series of subroutines design to verify the compliance of solutions to constraints. This heuristic algorithm yields cable layouts ranging from “good” to “acceptable”, with several test cases revealing sub-optimal routings identifiable by mere visual inspection. Fortunately, these sub-optimal routes do not appear in all design layouts, with the frequency of recurrence having a relation with the complexity of the layout and the number of turbines (in other words, less complexity means less likelihood of having sub-optimal routes). A post-processing manual correction can be performed on the resulting cabling layout to approximate the solution to the true optimum, indicating that this drawback of the POS1 can be easily addressed for small and medium wind farms where visual inspection reveals sub-optimal routes.

The second main algorithm implemented was the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) which has been reported in the literature as being a state-of-the-art heuristic approach for multivariable optimization problems. Furthermore, a previous successful application of the CMA in the turbine siting problem by Rodrigues et.al (36) served as a reassurance criteria for the selection of this algorithm. The CMA is capable of producing wind farm layouts that respect the imposed area constraints, displaying rapid convergence towards an asymptotic LPC value. From several optimizations it was concluded that the turbine siting problem as defined here doesn’t appear to have a single optimal solution, with a variety of layouts and wind farm LPC values resulting from repeated runs of the heuristic algorithm under the same design conditions and parameters. A comparison of optimized layouts against the initial guesses for the turbine positions revealed a heavy dependence on the solution on such a starting condition, with most turbines being displaced short distances before being almost “fixed” to a final position, even after several iterations of the CMA algorithm. This result, while not an indication of an inability to achieve converging layouts by the algorithm, does raise several questions regarding additional strategies and modifications to the solution approach that may be used to circumvent this problem.

Regarding time performance, while the CMA uses complicated data structures (mainly matrices and multiple poly-dimensional arrays) it was determined that the evolutionary strategy is not the bottleneck of the optimization strategy. The wake analyst was identified as the main bottleneck, mainly due to the multiple nested loops in its structure that result in a $O(n^2)$ time complexity with the number of turbines in the farm. The presence of such a bottleneck tremendously hinders the computational performance of the design tool, making the design of large wind farms (>50 turbines) a rather unpractical task. Nevertheless, as a tool for producing feasible optimized layouts the CMA implementation is considered a success and its application to solving the turbine siting problem is strongly encouraged.

In retrospect, the project objectives were met and the development of a solid base on which to base further work on the MZ tool was achieved. While there are still several improvements and extensions that need to be added to the tool to make it a truly powerful wind farm designer that is as close of a reflection of the real problem as possible, the ability to generate non-square layouts hereby developed is a substantial step in this direction.

9.2. Future Work

This project leaves a solid working framework on which to base future developments and further extend the capabilities of the MZ tool. The following is a list of possible issues that could be addressed as part of subsequent projects:

- The list of high-priority functional requirements to be addressed serves as a good guide to determine possible “next steps”. For example, providing the user with the choice of more than one objective function besides LPC is a good way to add functionality to the tool, providing the user with more freedom and possible outcomes and designs depending on his/her specific priorities.
- Extending the capabilities of the financial module is not an extremely complicated extension that could potentially provide valuable information for investors, such as wind farm NPV, IRR, cost-benefit ratio and other similar parameters.
- The topic of uncertainty in prediction of energy yield is a multi-layered problem that is not yet fully understood and still lacks standardized methods of estimation. However, a “crude” uncertainty module can be already developed with existing knowledge (some of the commercial wind farm design software analyzed possess such

capabilities); this addition combined with the extended financial assessment capabilities suggested in the previous point could greatly increase the attractiveness of the MZ tool not only as a software for designing but also to aid in selling a project.

- In order to produce more realistic wind farm designs the possibility to include non-uniform subsea conditions must be addressed. The inclusion of such models will not only result in a more versatile support structure designer (opening the possibility to even having more than one type of structure per wind farm) but also in a more challenging cable design problem that will have to deal with bathymetry and obstacles.
- There are several ways to improve the cable layout design. One aspect that remains to be addressed is enabling the POS1 algorithm to trace non-linear routes between points, thereby making the cables respect the area constraints as well. However, the chief recommendation would be to upgrade the POS1 to its more sophisticated version (POS2) and couple it with the RouteOpt extension to reduce the number of sub-optimal routes present in the final design. The POS2 shares most of its structure with the already implemented POS1, meaning that this shouldn’t be a very complex extension.
- The original motivation to generate wind farm layouts that respected certain rules of aesthetics was ultimately not addressed as part of this project, due mainly to this requirement ranking lower in the priority list than the treated problems. However, it remains an interesting problem to solve, that could be coupled with an option to enable the MZ tool to generate symmetric optimized layouts.
- Finally, if a host of different extensions are to be developed for the MZ tool, it is recommended that a more robust and capable graphical user interface (GUI) is developed. This addition will increase dramatically the value of the tool in the eyes of a developer, while also opening the possibility of the tool to be introduced to other types of users.

10. References

1. **European WInd Energy Association (EWEA).** *The European offshore wind industry - Key Trends and Statistics 2012.* s.l. : EWEA, 2013.

2. **Zaayer, M B.** *Great expectations for offshore wind turbines: Emulation of wind farm design to anticipate their value for customers.* Delft : DUWIND, 2013. ISBN 978-90-5335-752-1.

3. **Aghabi Rivas, Rajai.** *Optimization of Offshore Wind Farm Layouts.* Lyngby : Department of Mechanical Engineering, DTU, 2007.

4. **Chapra, Steven and Canale, Raymond.** *Numerical methods for engineers.* New York : McGraw Hill, 2010.

5. **Lucifero.** K-means clustering using Lloyd's algorithm. [Online] [Cited: 09 October 2013.] http://www.lucifero.us/toolbox/kmeans/.

6. *Offshore wind farm layout optimization using mathematical programming techniques.* **Pérez, B, Mínguez, R and Guanche, R.** 53, 2013, Renewable Energy, pag. 389-399.

7. *Iterative non-deterministic algorithms in on-shore wind farm design: a brief survey.* **Khan, Salman and Rehman, Shafiqur.** 19, 2013, Renewable and Sustainable Energy Reviews, págs. 370-384.

8. *Optimization of wind turbine positioning in large wind farms by means of a genetic algorithm.* **Mosetti, G, Poloni, C and Diviacco, B.** 1, 1994, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 51, pag. 105-116.

9. *Placement of wind turbines using genetic algorithms.* **Grady, S A, Hussaini, M Y and Abdullah, M M.** 30, 2005, Journal of Renewable Energy, pag. 259-270.

10. *An evolutive algorithm for optimal wind farm design.* **Mora, J, Baron, J, Santos, J, and Payan, J.** 70, 2007, Neurocomputing, pag. 2651-2658.

11. *Distributed genetic algorithm for optimization of wind farm annual profits.* **Huang, H.** 2007, Proceedings of the IEEE international conference intelligence system applied to power systems, pag. 1-6.

12. *Efficient hybrid distributed genetic algorithms for wind turbine placement in large wind farms.* **Huang, H.** 2009, Proceedings of the IEEE international symposium industrial electronics, pag. 2196-2201.

13. *Optimal positioning on wind turbines on Gokceada using multi-objective genetic algorithm.* **Sisbot, S, Turgut, O, Tunc, M, and Camdali, U.** 4, 2009, Wind Energy, Vol. 13, pag. 297-306.

14. *Optimization of wind farm turbines layout using an evolutive algorithm.* **González, J, Santos, J, and Payan, J.** 35, 2010, Journal of Renewable Energy, pag. 1671-1681.

15. **Jensen, N O.** *A Note on Wind Generator Interaction.* Roskilde : Risø National Laboratory, DK-4000, 1983.

16. *Wind farm optimal design including risk.* **González, J, Santos, J and Payan, M.** 2010, Proceedings of IEEE international symposium modelling electric power systems, pag. 1-6.

17. *Optimization of wind farm layout including decision making under risk.* **González, J, Santos, J and Payan, M.** 1, 2012, IEEE Systems Journal, Vol. 6, pag. 94-102.

18. *Wind farm layout optimization considering energy generation and noise propagation.* **Kwong, W, Zhang, P, Romero, D, Moran, J, Morgenroth, M, and Amon, C.** 2012, Proceedings of the ASME 2012 international des engineering technology conference & computing and information engineering conference, pag. 1-10.

19. *Optimal micro-siting of wind turbines by genetic algorithms based on improved wind and turbine models.* **Wan, C, Wang, J, Yang, G, and Zhang, X.** 2009, Proceedings of the 48th IEEE conference decision control, pag. 5092-5096.

20. *Modeling and simulation of optimal wind turbine configurations in wind farms.* **Wang, F, Liu, D and Zeng, L.** 2009, Proceedings of the IEEE world non-grid-connected wind power energy conference, pag. 1-5.

21. *Study on computational grid in placement of wind turbines using genetic algorithms.* **Wang, F, Liu, D and Zeng, L.** 2009, roceedings of the IEEE world non-grid-connected wind energy conference, pag. 1-4.

22. *Linear wind farm layout optimization through computational intelligence.* **Herbert-Acero, J, Franco-Acevedo, J, Valenzuela-Rendón, M, and Proebst-Oleszewski, O.** 2009, Proceedings of the Mexican international conference on artificia lintelligence, Lecture notes in artificial intelligence, pag. 692-703.

23. *New approach on optimization in placement of wind turbines within wind farm by genetic algorithms.* **Emami, A and Noghreh, P.** 25, 2010, Journal of Renewable Energy, pag. 1559-1564.

24. **Mittal, A.** *Optimization of the layout of large wind farms using a genetic algorithm.* Case Western Reserve University. 2010.

25. *Design of wind farm layout for maximum wind energy capture.* **Kusiak, A and Song, Z.** 35, 2010, Journal of Renewable Energy, pag. 685-694.

26. *CHC and SA applied towind energy optimization using real data.* **Bilbao, M and Alba, E.** 2010, Proceedings of the IEEE conference on evolutionary computation, pag. 1-8.

27. *Fuzzy multi-objective genetic algorithms for wind farm layout optimization.* **Khan, S and Rehman, S.** In Preparation.

28. *Optimization of wind farm layout.* **Rasuo, B and Bengin, A.** 38, 2010, FME Transformers, pag. 107-114

29. *On aerodynamic optimization of wind farm layout.* **Rasuo, B and Bengin, A.** 1, 2010, Proceedings of the applied mathematics and mechanics, Vol. 10, pag. 539-540.

30. **Tesauro, A, Réthoré, P E and Larsen, G C.** *State of the art of wind farm optimization.* Roskilde : DTU Wind Energy, Riso Campus.

31. *A novel method for optimal placing wind turbines in a wind farm using particle swarm optimization (PSO).* **Rahmani, R, Khairuddin, A, Cherati, S, and Pesaran, H.** 2010, Proceedings of the IEEE international conference on power engineering, pag. 134-139.

32. *Optimal micro-siting of wind farms by particle swarm optimization.* **Wan, C, Yang, G, Li, X, and Zhang, X.** 2010, Proceedings of the international conference on swarm intelligence, Lecture notes in computer science , pag. 198-205.

33. *Exploring key factors influecing optimal farm design using mixed-discrete particle swarm optimization.* **Chowdbury, S and Zhang, J.** 2010, Proceedings of the 13th AIAA/ISSMO multidisciplinary analysis and optimization conference, pag. 1-16.

34. *Unrestricted wind farm layout optimization UWFLO: investigating key factors influecing the maximum power generation.* **Chowdbury, S, Zhang, J, Messiac, A, and Castillo, L.** 38, 2012, Journal of Renewable Energy, pag. 16-30.

35. *GA and PSO applied to wind energy optimization.* **Bilbao, M and Alba , E.** 2009, XV Congreso Argentino de Ciencias de la Computación..

36. **Rodrigues, S, Bauer, P and Pierik, J.** *Modular Approach for the Optimal Wind Turbine Micro-Siting Problem through CMA-ES Algorithm.*

37. *A fast and effective local search algorithm for optimizing the placement of wind turbines.* **Wagner, M, Day, J and Neumann, F.** 2012, Renewable Energy, Vol. 51, pag. 64-70.

38. **AWS Truepower.** Solutions: openWind Enterprise. [Online] [Cited: 19 October 2013.] http://www.awstruepower.com/solutions/products/openwind/.

39. *Seeding evolutionary algorithms with heuristics for optimal wind turbines positioning in wind farms.* **Saavedra-Moreno, B, Salcedo-Sanz, S, Paniagua-Tineo, A, Prieto, L, and Portilla-Figueras A.** 36, 2011, Renewable Energy, pag. 2838-2844.

40. *TopFarm: Multi-fidelity optimization of offshore wind farm.* **Réthoré, P, Larsen, G, Buhl, T, Larsen, T, and Madsen H.** Roskilde : s.n., Wind Energy Division, Riso DTU- National Laboratory for Sustainable Energy.

41. **Katic, I, Jensen, N O and Hojstrup, J.** *A Simple Model for Cluster Efficiency.* Rome : European Wind Energy Association Conference and Exhibition, 1986.

42. **González, Julian E.** *Developing an Efficient Algorithm to Assess Wind Turbine Wake Effects.* Delft : System Integration Project II, MSc Sustainable Energy Technology, 2013.

43. *Offshore Wind Farm Layout Optimization (OWFLO) Project: Preliminary Results.* **Elkinton, C, Manwell, J and McGowan, J.** University of Massachusetts.

44. *Optimal placement of wind turbines in a wind park using Monte Carlo simulation.* **Marmidis, G, Lazarou, S and Pyrgioti, E.** 33, 2008, Renewable Energy, pag. 259-270.

45. *Ecologic and economic cost-benefit analysis of offshore wind energy.* **Snyder, B and Kaiser, M.** 34, 2009, Renewable Energy, pag. 1567-1578.

46. **Manwell, J, McGowan, J and Roger, A.** *Wind Energy Explained: Theory, Design and Application.* s.l. : Wiley & Sons., 2009..

47. **Birkhoff, George D.** *Aesthetic Measure.* s.l. : Cambridge Massachusets' University Press, 1933.

48. *Conceptualizing Birkhoff's Aesthetic Measure Using Shannon Entropy and Kolmogorov Complexity.* **Rigau, J, Feixas, M and Sbert, M.** 2007, Computational Aesthetics in Graphics, Visualization and Imaging.

49. **Schachner, J.** *Power connections for offshore wind farms.* Delft : TU Delft Diploma Thesis, 2004.

50. **GL Garrad Hassan.** WindFarmer. *Modules.* [Online] [Cited: 19 October 2013.] http://www.gl-garradhassan.com/en/software/windfarmer/16206.php.

51. **EMD International A/S.** WindPro. *Modules.* [Online] [Cited: 19 October 2013.] http://www.emd.dk/WindPRO/Modules/.

52. **ReSoft.** WindFarm from ReSoft. *Details.* [Online] [Cited: 19 October 2013.] http://www.resoft.co.uk/English/html/details.htm.

53. **Striedinger, A.** *Untitled Master Thesis.* Delft : s.n., 2013.

54. **Arends, E, Jaspers, M and van der Bilt, S.** *Aanvraag Watervergunning Offshore Windpark Q4 West.* Hengelo : Pondera Consult, 2013. 712006.

55. **Jarquin, A.** *Offshore Design Considerations.* Delft : Offshore Wind Farm Design lecture notes, 2012.

56. **DONG Energy.** Anholt Offshore Wind Farm Newsteller, January 2013. [Online] January 2013. [Cited: 4 December 2013.] http://www.dongenergy.com/anholt/EN/News/anholt_nyheder/News/Pages/AnholtOffshoreWindFarm-Newsletter-January2013.aspx.

57. *Wave models for offshore wind turbines.* **Argawal, P and Manuel, L.** Reno : s.n., 7-10 January 2008, 46th AIAA Aerospace Sciences Meeting and Exhibit, Vols. 2008-1336.

58. **Bierbooms, W.** *Offshore wind farm aspects.* Delft : Offshore wind farm design lecture slides, 2012.

59. *Uncertainties in the design of support structures and foundations for offshore wind turbines.* **Negro, V, López-Gutiérrez, J, Dolores, M, and Matunano, C.** March 2014, Renewable Energy, Vol. 63, pag. 125-132.

60. **cmelan.** NEOS Guide. *Multiple Traveling Salesman Problem (mTSP).* [Online] 18 June 2012. [Cited: 27 November 2013.] http://www.neos-guide.org/content/multiple-traveling-salesman-problem-mtsp.

61. *The multiple traveling salesman problem: an overview of formulations and solution procedures.* **Bektas, Tolga.** 34, 2006, Omega, pag. 209-216.

62. **Department of Computer Science and Engineering, Hong Kong University of Science and Technology.** Lecture 7: Minimum Spanning Trees and Prim's Algorithm. [Online] [Cited: 5 December 2013.] http://www.cse.ust.hk/~dekai/271/notes/L07/L07.pdf.

63. *The Offshore Wind Farm Array Cable Layout Problem - A Planar Open Vehicle Routing Problem.* **Bauer, J and Lysgaard, J.** 2013, Invited abstract in session WC-17: Optimization problems in the offshore wind industry, stream Maritime Transportation.

64. **Hansen, Nikolaus.** *The CMA Evolution Strategy: A Tutorial.* 2011.

65. —. *The CMA Evolution Strategy: A Comparing Review.* s.l. : CoLab Computational Laboratory, ETH Zurich; ICoS Institute of Computational Science, ETH Zurich.

66. **Smith, Alice y Coit, David.** *Penalty Functions.* Pittsburgh : Oxford University Press and Institute of Physics Publishing, 1995.

67. **Sanderse, B.** *Aerodynamics of Wind Turbine Wakes: Literature Review.* s.l. : Energy Research Centre of the Netherlands (ECN), 2009. ECN-E--09-016.

68. *The effect of a giant wind farm on precipitation in a regional climate model.* **Fiedler, B and Bukovsky, M.** 6, 2011, Environmental Research Letters. 045101.

69. *Potential climatic impacts and reliability of large-scale offshore wind farms.* **Wang, C and Prienn, R.** 6, 2011, Environmental research letters. 025101.

70. *Environmental impact of wind energy.* **Saidur, R, Rahim, N, Islam, M, and Solangi, K.** 15, Renewable and Sustainable Energy Reviews : s.n., 2011, pag. 2423-2430.

71. *Corrigendum: Environmental implications of large-scale adoption of wind power: a scenario-based life cycle assessment.* **Arvesen, A and Hertwich, E.** 7, 2012, Environmental research letters. 039501.

72. *Environmental impact of wind energy.* **Mann, J and Teilmann, J.** 8, 2013, Environmental Research Letters. 035001.

73. *Short-term ecological effects of an offshore wind farm in the Dutch coastal zone; a compilation.* **Lindeboom, H, Bergman, M, Bouma, S, Brasseur, S, Daan, R, Fijn, R, de Haan, D, Dirksen, S, van Hal, R, Hille Ris Lambers, R, ten Hofstede, R, Krijgsveld, K, Leopold, M, and Schiedat, M.** 6, 2011, Environmental Research Letters. 035101.

74. *Harbour porpoises (Phocoena phocoena) and wind farms: a case study in the Dutch North Sea.* **Scheidat, M, Tougaard, J, Brasseur, S, Carstensen, J, van Polanen Petel, T, Teilmann, J, and Reijnders, P.** 6, 2011, Environmental Research Letters. 025102.

75. *Investigation of the bird collision risk and the responses of harbor porpoises in the offshore sind farms Horns Rev, North Sea and Nysted, Baltic Sea in Denmark.* **Diederichs, A, Hennig, V and Niels, G.**

76. *Negative long term effects on harbour porpoises from a large scale offshore wind farm in the Baltic - evidence of slow recovery.* **Teilmann, J and Cartensen, J.** 7, 2012, Environmental Research Letters. 045101.

77. *Wind energy development and its environmental impact: a review.* **Leung, Dennis and Yang, Yuan.** 16, 2012, Renewable and Sustainable Energy Reviews, pag. 1031-1039.

78. *Understanding public responses to offshore wind power.* **Haggett, Claire.** 39, 2011, Energy Policy, pag. 503-510.

79. *Preference of coastal zone user groups regarding the siting of offshore wind farms.* **Ladenburg, J and Dubgaard, A.** 52, 2009, Ocean and coastal management, pag. 233-242.

80. *Visual assessment of off-shore wind turbines: The influence of distance, contrast, movement and social variables.* **Bishop, I and Miller, D.** 32, 2007, Renewable Energy, pag. 814-831.

81. **van Bussel, G.** *Operation and Maintenance.* Delft : Offshore wind farm design lecture slides, 2012.

82. *A Note on Two Problems in Connexion with Graphs.* **Dijkstra, E W.** 1, 1959, Numerische Mathematlk, pag. 269-271.

83. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths.* **Hart, P, Nilsson, N and Raphael, B.** 2, 1968, IEEE Transactions of Systems Science and Cybernetics, Vols. SSC-4.

84. *Integer linear programming formulations of multiple salesman problems and its variations.* **Kara, I and Bektas, T.** 174, 2006, European Journal of Operational Research, pag. 1449–1458.

Appendixes

Appendix A: Lists of Functional Requirements

Category A Requirement	S	R	F	P	Pr
Include a built-in help menu or manual to aid the user.	V	5	5	0	**10**
Achieve a high-quality solution in a reasonable amount of time.	II	5	5	5	**15**
Include an ample catalogue of commercially available wind turbines for the user to choose from.	I, V	4	4	4	**12**
Allow the user to use create and use custom wind turbines.	I	4	5	4	**13**
Allow the user to use and analyze personal wind data.	I, V	4	4	3	**11**
Automatically obtain, import or otherwise allow for the input of the wind speed distribution.	I	4	4	3	**11**
Ability to either receive the power curve as input or otherwise generate an approximation to it.	I, IV	5	5	5	**15**

Table A 1. Type A (software requirements and inputs) functional requirements.

Category B Requirement	S	R	F	P	Pr
Ability to generate optimized symmetrical (or slightly organized) turbine layouts.	I, II, III	5	5	5	**15**
Ability to generate optimized non-organized layouts.	I, II, III, IV	5	5	5	**15**
Restrict available turbine siting area due to various constraints.	I, II, III	5	5	5	**15**
Determine the optimal number of wind turbines to be installed (make it an output rather than an input).	IV	3	4	4	**11**
Apply a constraint of minimum turbine separation.	II, III	5	5	5	**15**

Table A 2. Type B (layout related requirements) functional requirements.

Category C Requirement	S	R	F	P	Pr
Develop and implement a more complex wake model, including factors such as surface interactions and meandering.	II	3	1	1	**5**
Develop and implement better cost models that offer a more detailed breakdown of every contributing factor.	II, IV	4	2	2	**8**
Include a robust wake model that can handle any layout.	I, IV	5	5	5	**15**
Provide the user the choice of at least two wake models.	I, II	2	3	3	**8**
Account for atmospheric turbulence effects.	I, II	3	4	2	**9**
Include models to determine hydrodynamic and aerodynamic loads.	I, IV	5	4	5	**14**
Ability to calculate energy yield considering topographic effects.	I	0	3	0	**3**
Enable yield calculations without accounting for wake effects (site conditions alone).	I	2	5	0	**7**
Have a fully implemented and functional wind rose.	I	5	4	4	**13**

Table A 3. Type C (modeling and calculations requirements) functional requirements.

Category D Requirement	S	R	F	P	Pr
Include wind flow modeling and visualization capabilities, for example through the use of WAsP.	I	3	1	0	**4**
Import and use worldwide maps, geographic data and layers.	I, V	4	2	0	**6**
Visualize all environmental impacts on neighboring zones in a map or 2D-plot.	I	1	1	1	**3**
Allow the use of scanned maps as the visual base (workspace).	I	4	3	2	**9**
Provide GIS-styled functionality and interface.	I	5	2	0	**7**
Visualize outputs of calculations in maps.	IV	5	2	1	**8**

Table A 4. Type D (graphics and user interface) functional constraints.

Category E Requirement	S	R	F	P	Pr
Include climatic impact (local and global) models and assessment capabilities.	II	2	3	3	**8**
Include a model to assess impacts on wildlife.	II	1	2	1	**4**
Visual impact assessment capabilities, either by ZVI, photomontages or rendering.	I	5	4	0	**9**
Include shadow flicker effects assessment and visualization.	I	0	3	0	**3**
Include noise impact assessment tools.	I	0	5	3	**8**
Ability to evaluate compliance with local noise limits.	I	0	5	2	**7**
Assess radar ceiling effects.	I	2	2	1	**5**

Table A 5. Type E (environmental aspects) functional requirements.

Category F Requirement	S	R	F	P	Pr
Develop and implement new objective functions which encompass all relevant problem variables and prove to yield good results.	II, V	5	3	4	**12**
Give the user a choice of at least two objective functions.	I	5	4	4	**13**
Implement a heuristic optimization algorithm.	II	4	4	5	**13**
Implement a local search optimization algorithm	II	4	4	5	**13**
Include COE/LPC as an objective function due to its known relevance to the problem, comprehensive formulation and acceptance by most users.	I, II	5	5	5	**15**
Develop a strategy that minimizes randomization errors that stem from the heuristic nature of the algorithms.	II	3	2	3	**8**
Ability to optimize for a site with variable water depth.	I, IV	4	3	3	**10**

Table A 6. Type F (optimization routine requirements) functional requirements.

Category G Requirement	S	R	F	P	Pr
Ability to evaluate financial performance indicators, such as NPV or IRR for the project.	I, IV	4	5	5	**14**
Include the option of defining custom financial parameters, such as interest and inflation rates.	V	4	5	5	**14**

Include a detailed installation cost module, explicitly stating its breakdown into components.	IV	3	4	4	**11**
Include a detailed O&M costs module, explicitly stating its breakdown into components.	IV	3	4	4	**11**
Allow the user to input and assess the effect of project lifetime on total costs.	IV	3	5	3	**11**
Implement a future value model for all project assets, to refine the overall cost model.	IV	2	4	3	**9**

Table A 7. Type G (optimization routine requirements) functional requirements.

Category H Requirement	**S**	**R**	**F**	**P**	**Pr**
Automated support structure design.	I	5	4	5	**14**
Provide the user a choice of at least two support structures (monopile and jacket).	V	3	3	4	**10**
Ability to calculate loads on the structure and automatically determine compliance with the IEC-61400 standard.	I, V	4	4	5	**13**

Table A 8. Type H (support structures) functional requirements.

Category I Requirement	**S**	**R**	**F**	**P**	**Pr**
Have the ability to visualize the optimized cable layout.	I, V	3	2	2	**7**
Implement an automated cable topology optimizer, closely tied to the layout optimization routine.	I, III, IV	5	4	5	**14**

Table A 9. Type I (electrical aspects) functional requirements.

Category J Requirement	**S**	**R**	**F**	**P**	**Pr**
Ability to calibrate tool outputs with real site data.	I	1	4	1	**6**
Determine optimal start-up and shut-down strategies based on the optimized wind farm.	I	1	3	1	**5**
Ability to analyze uncertainty of the energy yield calculation.	I, V	5	4	4	**13**
Include tools that allow for performance assessment of already existing projects.	I	1	4	0	**5**
Couple a wind farm controller optimizer to the existing routine.	V	4	2	3	**9**

Table A 10. Type J (miscellaneous) functional requirements.

Appendix B: Subroutines for Cabling Algorithm POS1

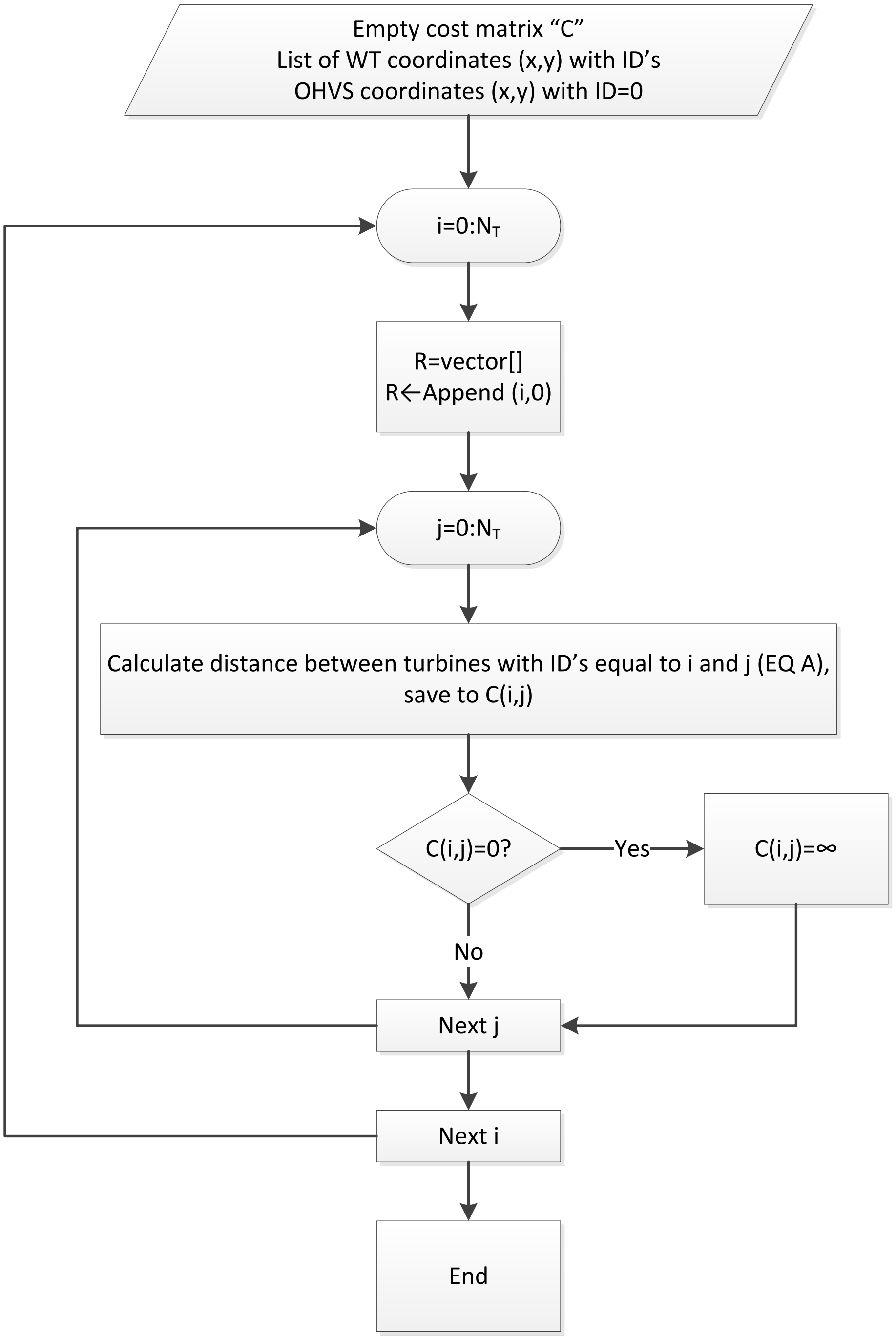

Empty cost matrix “C”
List of WT coordinates (x,y) with ID’s
OHVS coordinates (x,y) with ID=0
i=0:N_T
R=vector[]
R←Append (i,0)
j=0:N_T
Calculate distance between turbines with ID’s equal to i and j (EQ A), save to C(i,j)
C(i,j)=0?
Yes
C(i,j)=∞
No
Next j
Next i
End

Figure B 1. Subroutine 1, set cost (distance) matrix.

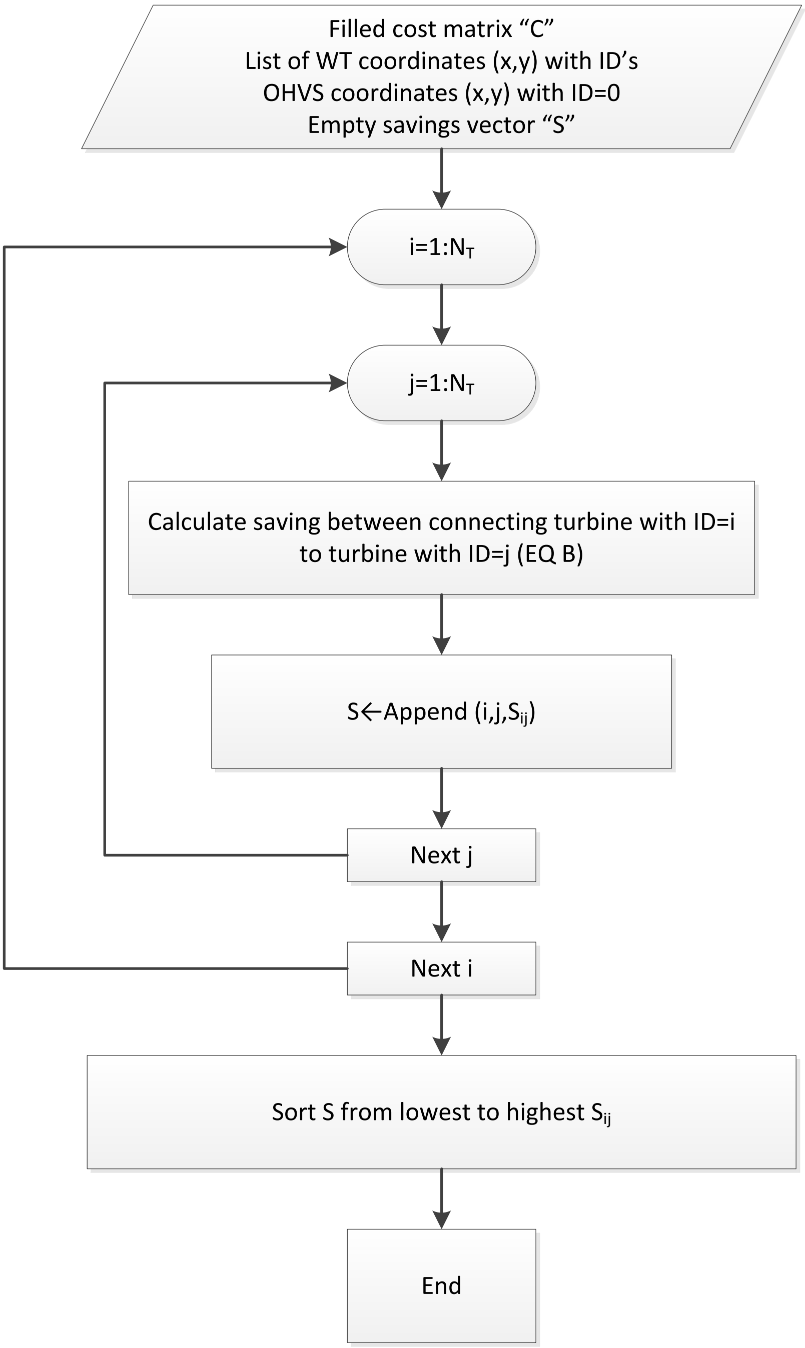

Filled cost matrix “C”
List of WT coordinates (x,y) with ID’s
OHVS coordinates (x,y) with ID=0
Empty savings vector “S”
i=1:N_T
j=1:N_T
Calculate saving between connecting turbine with ID=i to turbine with ID=j (EQ B)
S←Append (i,j,S_{ij})
Next j
Next i
Sort S from lowest to highest S_{ij}
End

Figure B 2. Subroutine 2, set and sort savings vector.

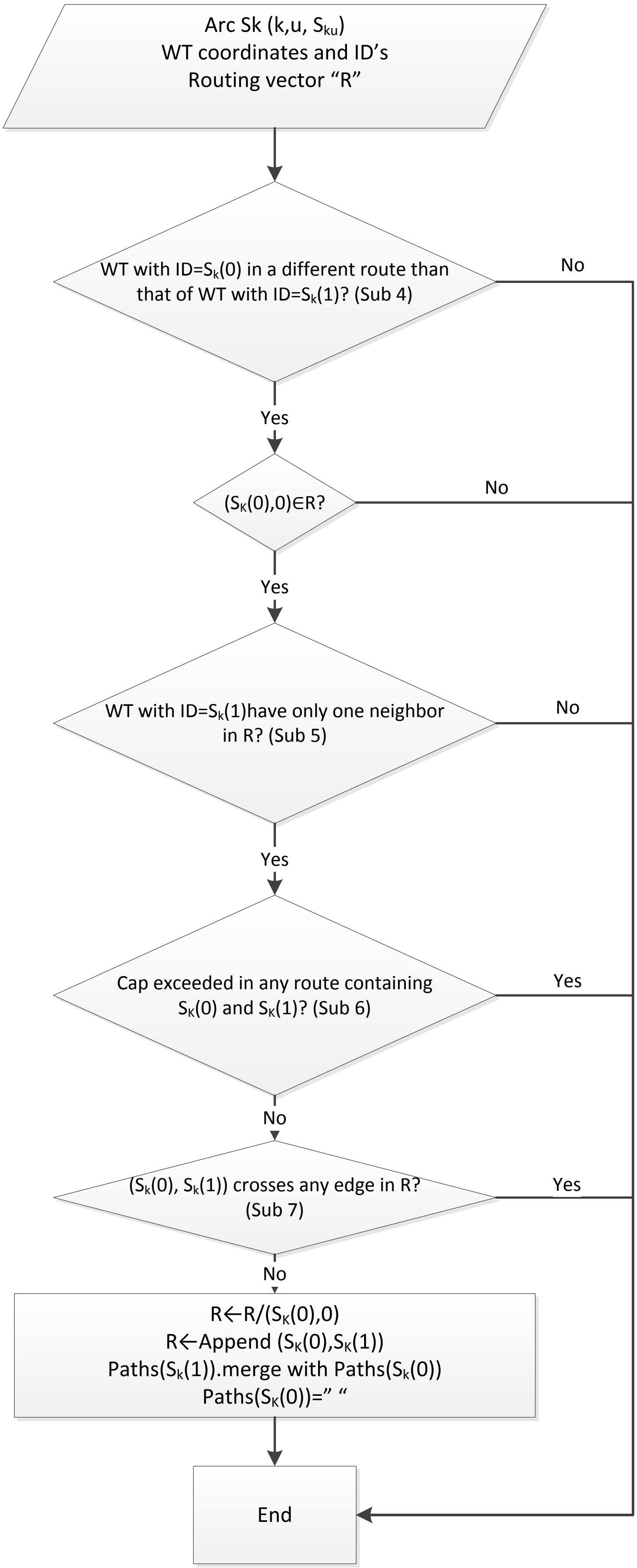

Arc Sk (k,u, S_{ku})
WT coordinates and ID's
Routing vector "R"
WT with ID=$S_k(0)$ in a different route than that of WT with ID=$S_k(1)$? (Sub 4)
No
Yes
($S_K(0)$,0)∈R?
No
Yes
WT with ID=$S_k(1)$have only one neighbor in R? (Sub 5)
No
Yes
Cap exceeded in any route containing $S_K(0)$ and $S_k(1)$? (Sub 6)
Yes
No
($S_k(0)$, $S_k(1)$) crosses any edge in R? (Sub 7)
Yes
No
R←R/($S_K(0)$,0)
R←Append ($S_K(0)$,$S_K(1)$)
Paths($S_k(1)$).merge with Paths($S_k(0)$)
Paths($S_K(0)$)=" "
End

Figure B 3. Subroutine 3, condition checks for every arc.

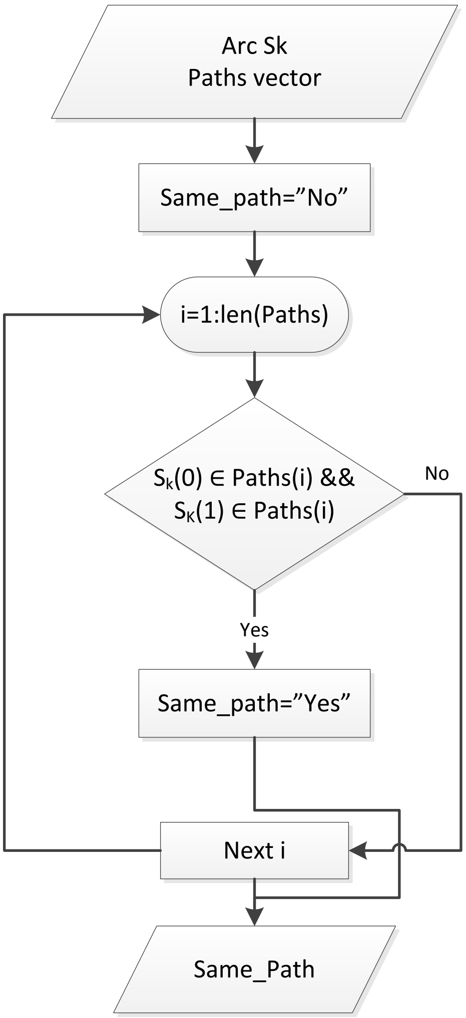


Figure B 4. Subroutine 4, determine if two turbines in an arc (k,u) are already in the same cabling route.

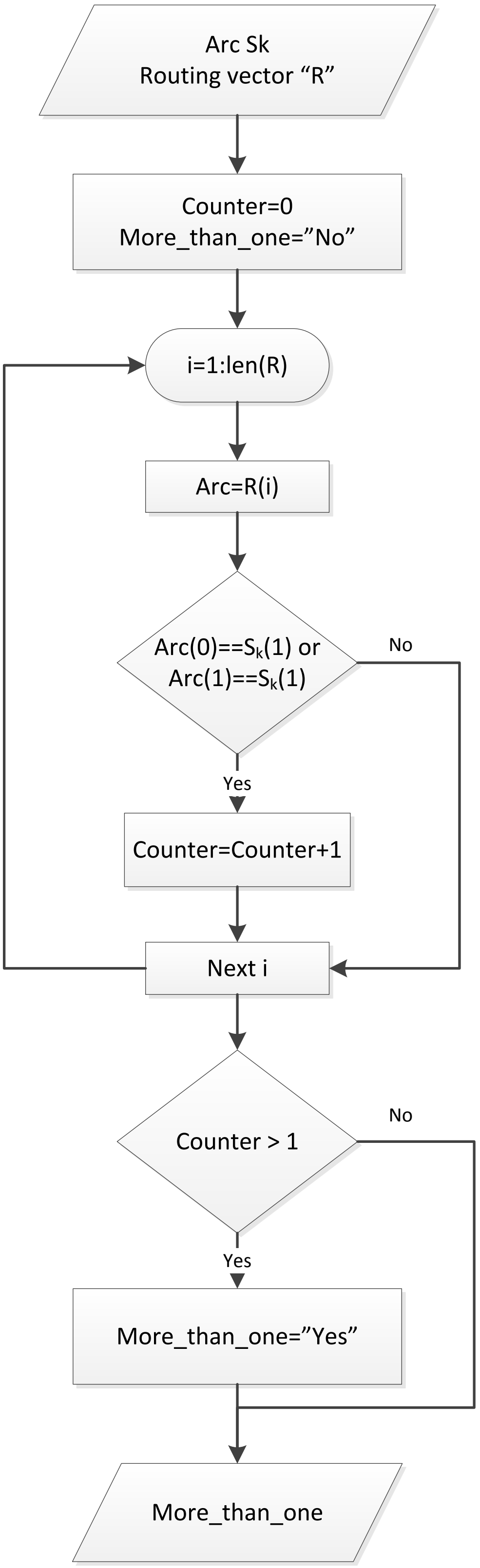


Figure B 5. Subroutine 5, determine if a turbine has more than one neighbor in the same route.

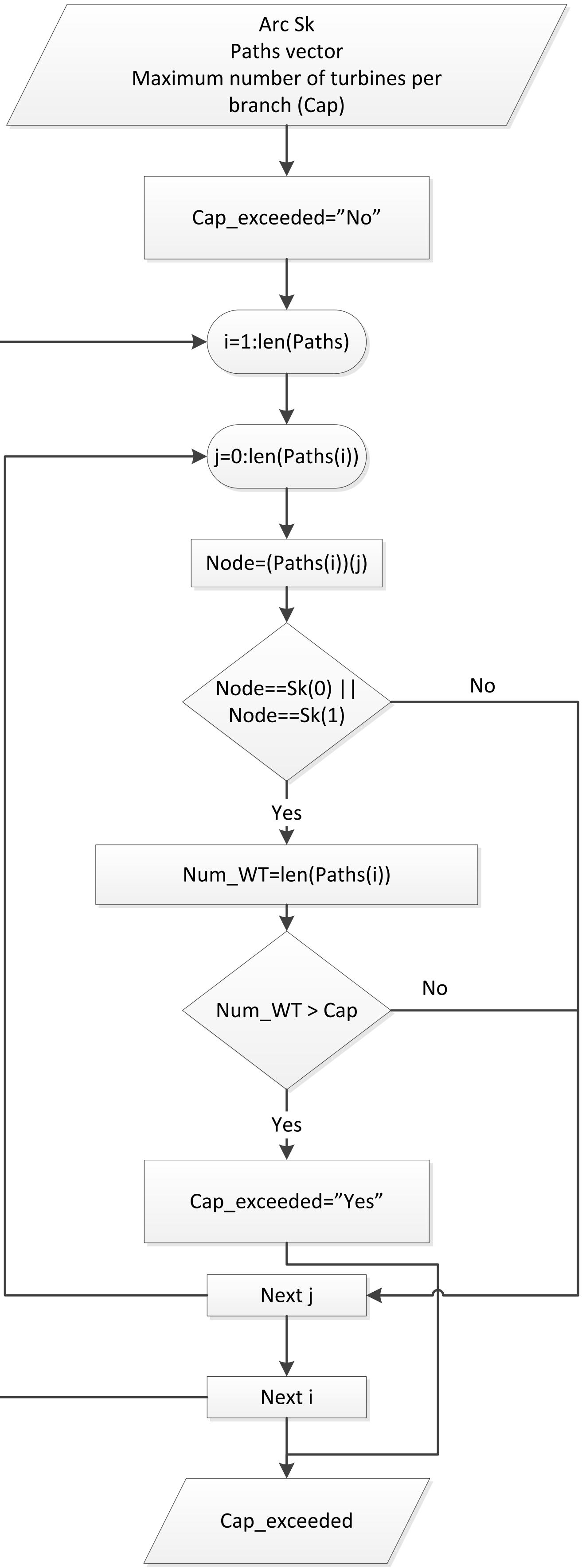


Figure B 6. Subroutine 6, check for exceeded capacity.

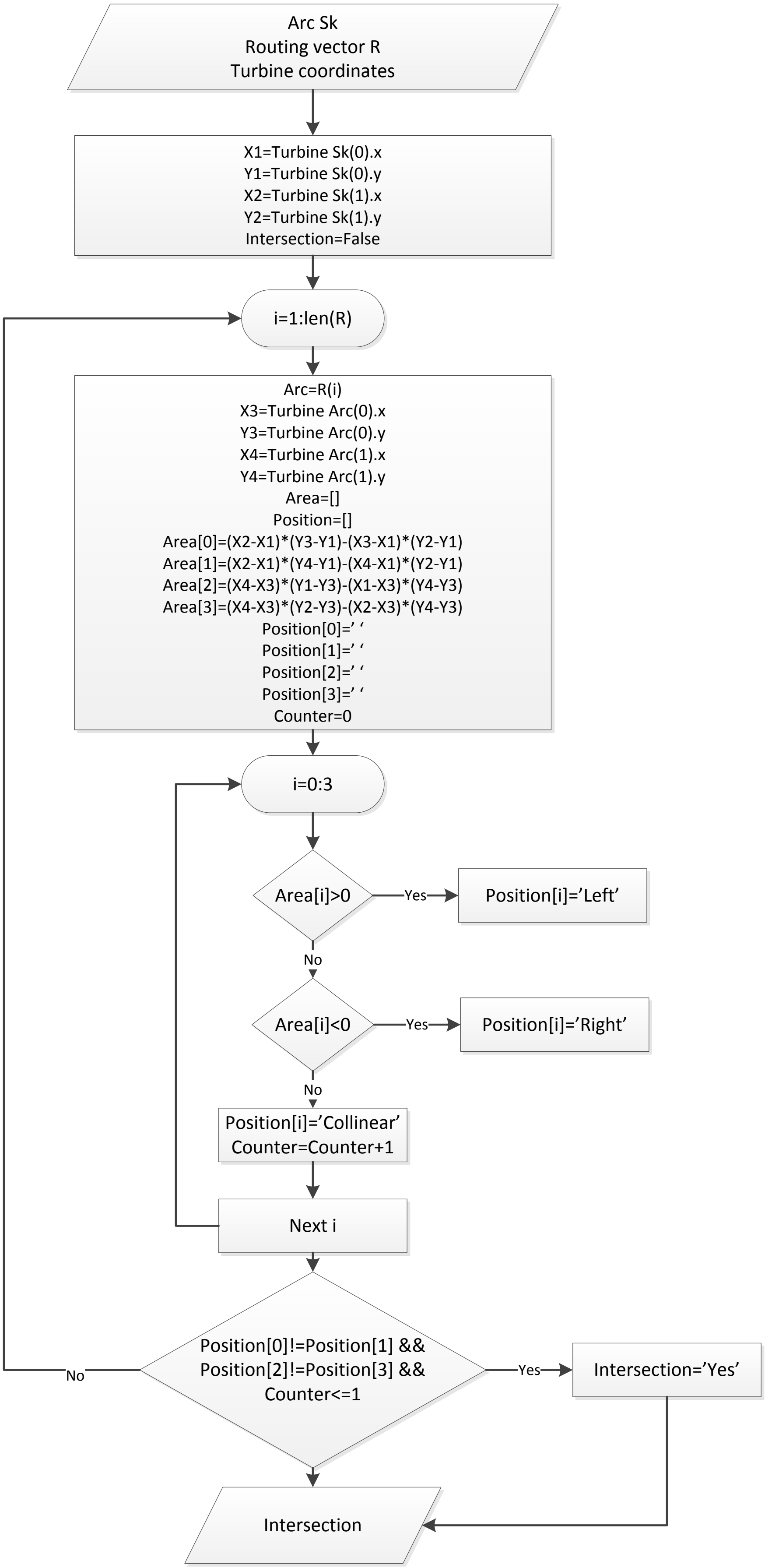


Figure B 7. Subroutine 7, determine if two segments of line intersect.

Appendix C: Guided Example of POS1

Although the work by Bauer et.al (63) contains the pseudo-code to implement POS1, it may be more helpful to illustrate the way it works by means of an example. Consider the case of five wind turbines and one OHVS with known coordinates; the goal is to generate the optimal cabling route that connects all turbines to the OHVS once and only once. Also, no single branch may have more than a specified number of turbines (capacity, or Cap), which in this example will be set to 3.

The optimal routing is defined by a so-called routing vector R, which is initially defined as follows:

$$R = [(1,0), (2,0), (3,0), (4,0), (5,0)]$$

Each element in the vector R is called an arc, which is the path created by connecting two turbines together or a turbine to the OHVS. Only arcs that are part of the routing vector are those that comprise the optimal solution. Therefore, initially, the cable routing looks as shown in Figure C 1. The star numbered “0” represents the OHVS, while all the circles are the turbines to be connected, each number representing their identification number (ID).

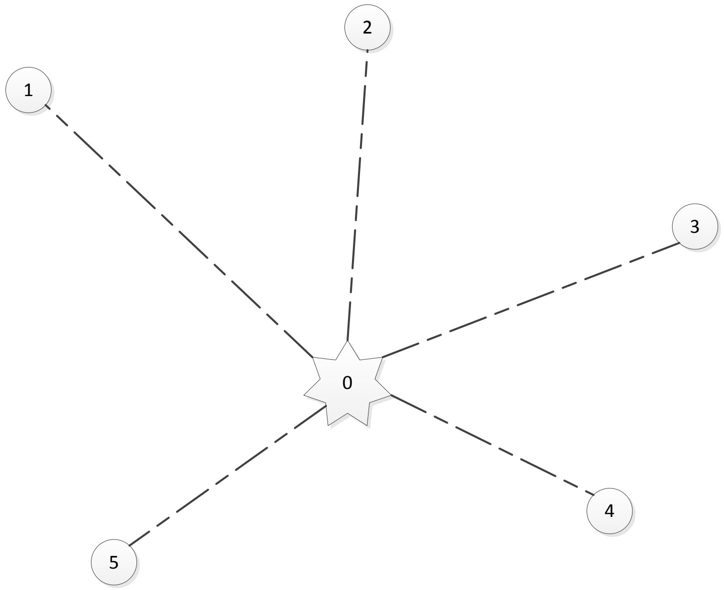


Figure C 1. Initial cabling routing as defined by the R vector.

Since the Cartesian coordinates of all six points are known the distance between every turbine and its neighbors may be calculated using simple trigonometry. The savings heuristic was originally defined in terms of travel costs between one point and the other, the general idea being the minimization of such costs in the final solution. Since the cabling problem deals with minimizing the cabling distance, cost and distance can be used interchangeably in the problem’s context. Thus, a cost (or distance) matrix is generated by filling the distances between the respective points. The matrix in EQU is an example of a cost matrix, using arbitrary values for the distances between points. It can be easily read as “from ROW to COLUMN”, for example, the distance from turbine 1 to turbine 3 is 29.

$$C = \begin{bmatrix} - & 28 & 31 & 20 & 25 & 34 \\ & - & 21 & 29 & 26 & 20 \\ & & - & 38 & 20 & 32 \\ & & & - & 30 & 27 \\ & & & & - & 25 \\ & & & & & - \end{bmatrix}$$

Notice that this matrix is both square and symmetric, since the distance from A to B is the same as B to A. Additionally, the diagonal is filled with hyphens instead of zeroes, with the hyphen representing an infinite distance.

The next step is calculating the savings matrix, with each saving S_{ku} representing how much cost is saved by merging two routes into a single one containing the arc (*k,u*).This saving is calculated according to the definition in EQ, in which the subindex *d* represents the OHVC, or point 0.

$$s_{ku} = C_{kd} - C_{ku}$$

The resulting savings matrix is not symmetric and has dimensions N_TxN_T. The purpose of calculating the savings achieved by merging for every pair of turbines is to allow the heuristic algorithm to choose the merge with the highest saving that doesn’t exceed the capacity per route, which will in turn result in an optimal route. Using the numerical values given by the cost matrix and applying the savings definition the resulting savings matrix is as follows:

$$S = \begin{bmatrix} - & 7 & -1 & 2 & 8 \\ 3 & - & -7 & 11 & -1 \\ -9 & -18 & - & -10 & -7 \\ -1 & 5 & -5 & - & 0 \\ 14 & 2 & 7 & 9 & - \end{bmatrix}$$

Once the savings matrix is calculated the corresponding arcs are sorted from highest to lowest saving, resulting in an arc vector A like this:

$$A = [(5,1),(2,4),(5,4),\dots,(3,1),(3,4),(3,2)]$$

Therefore, merging routes using the arc (5,1) creates the highest saving, while doing so with (3,2) generates the lowest and thus it is the least preferred routing option. The final step is to iterate over vector A, and for every arc evaluate the following conditions:

- Are *k* and *u* in different routes?
- Is the arc (*k,d*) ∈ R?

- Point *u* has only one element in R?
- The total number of points in the paths containing *k* and *u* does not exceed Cap?
- The arc (*k*,*u*) does not cross any arc in R?

If all five conditions are satisfied then the arc is added to the routing vector R, and one of the initialization arcs is removed. The first arc in A, namely, (5,1), satisfies all the conditions, thus R is altered as follows:

$$R \leftarrow R\ (k,d) \cup (k,u) = [(1,0),(2,0),(3,0),(4,0),(5,1)]$$

After all arcs in A have been evaluated the resulted routing vector is:

$$R = [(2,0),(3,0),(5,1),(1,2),(4,3)]$$

This routing is illustrated in Figure C 2.

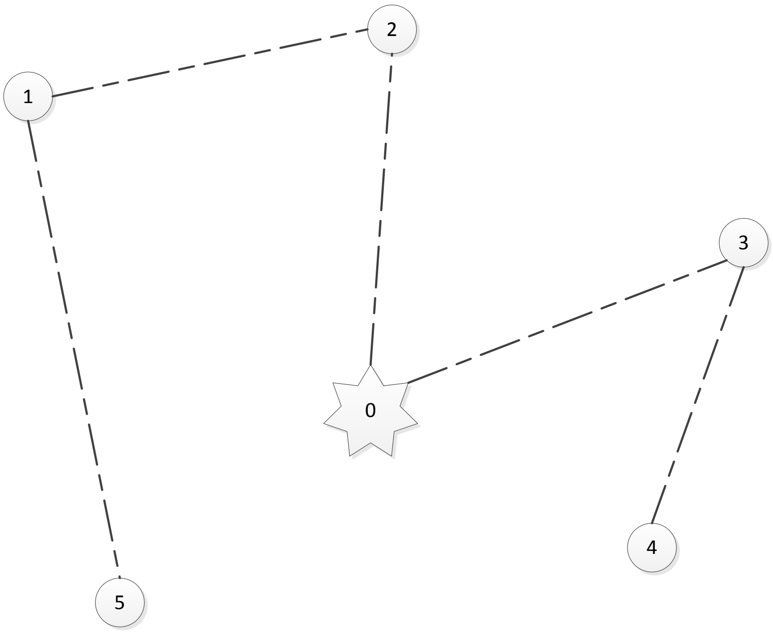


Figure C 2. Final routing for the example, two routes, no route may hold more than three turbines.