Surrogate-Assisted Evolutionary Algorithms for Wind Farm Layout Optimisation Problem

A thesis
submitted in partial fulfillment
of the requirements for the degree
of

Master of Science (Research)

at

The University of Waikato

by

Chen Zheng

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2016

Abstract

Due to the increasing need for computationally expensive optimisation in many real-world applications, surrogate-assisted evolutionary algorithms have attracted growing attention. In the literature, surrogate-assisted evolutionary approaches have been successful in highly computational expensive optimisation problems. However, surrogates have not been used with the Wind Farm Layout Optimisation Problem (WFLOP) before.

In this work, an evolutionary approach using surrogate modelling techniques to reduce the computational cost of the WFLOP is studied. The WFLOP mainly focuses on finding the optimal geographical placement of wind turbines within a wind farm in order to maximise power generation. But evaluating wind farm layout is very computationally expensive. The purpose of using surrogates is to approximate the real evaluation function of an evolutionary algorithm, but the surrogates can be computed more efficiently. The aim of this study is try to discover whether the surrogate-assisted evolutionary approach is effective on the WFLOP or not.

An analytical wake model from the literture is used to calculate the velocity deficits in the downstream generated by individual turbines. A set of initial offline experiments was conducted based on a dataset of wind farm layouts sampled from the space of all layouts, using biased random walk. These experiments were designed to discover which features lead to construction of an accurate surrogate model. According to the results of these experiments, polar coordinates (sorted according to distance) as features are selected for learning. A multilayer per-

ceptron (MLP) neural network and a tree-based regression model (M5P) are chosen as the surrogate models to approximate the real fitness function in conjunction with an (μ, λ) evolutionary strategy. Two previously presented BlockCopy operators are used in the evolutionary strategy. The surrogate models are managed using a modified version of the Preselection strategy and the Best strategy.

Our evaluation used four benchmark wind farm scenarios with dimensionality ranging from 200 to 1420 dimensions. The evaluation results show that our preliminary MLP and M5P surrogate models did not improve the optimisation results over traditional evolutionary strategies due to scalability issues. The scalability is a known weakness of many surrogate-assisted evolutionary approaches for the reason that most of them are designed for low-dimensionality problems. However, the research should continue on this topic because of its importance to renewable energy.

Acknowledgements

First of all, I would like to express my deepest appreciation and gratitude to my supervisor Dr. Michael Mayo for his expert guidance, understanding and encouragement throughout my study and research. He read my numerous revisions and he was always available for my questions. Without his incredible patience and full support, the completion of this study would not have been possible. It is a great honour to work under his supervision.

Next I would like to thank my parents for their love and support over the years. They have always been there for me and I am thankful for everything they have helped me achieve.

I am also grateful to all the fellow graduate students in the Machine Learning Lab at the Department of Computer Science, the University of Waikato. Especially a thank you to my friend and class fellow Bingyuan Liu for always listening and giving me words of encouragement.

Last but not least, I am thankful to all my friends for helping me survive all the stress from this year. I have no valuable words to express my thanks, but my heart is still full of the favours received from every person.

Contents

1 Introduction 1

2 Wind Farm Layout Optimisation Problem 5
2.1 Wind Farm Layout Example 5
2.2 Wind Farm Wake Effect 7
2.2.1 Wind Turbine Characteristics 7
2.2.2 Surface Roughness 10
2.2.3 Wind Modelling 11
2.2.4 Wake Effect Modelling 13
2.3 Wind Turbine Energy Output 18
2.4 WFLOP Objective Functions 19
2.5 Wind Farm Design Tools 22
2.6 Variants of the WFLOP 24
2.7 Computational Complexity of WFLOP 26

3 Evolutionary Algorithms 29
3.1 Genetic Algorithms 30
3.2 Evolutionary Strategies 31
3.3 Evolutionary Strategies with Elitism 33
3.4 Selection, Mutate and Crossover 34
3.5 Crossover vs. Mutation 35
3.6 Exploration vs. Exploitation 37

3.7 Evolutionary Algorithm Applications for WFLOP 38

4 BlockCopy Operators for the WFLOP 43
4.1 BlockCopy Mutation Operator 44
4.2 BlockCopy Crossover Operator 46

5 Surrogate-Assisted Evolutionary Optimisation 49
5.1 Overview 49
5.2 Surrogate Modelling Techniques 51
5.2.1 Quadratic Response Surface Model 52
5.2.2 Kriging Model 55
5.2.3 Artificial Neural Networks 57
5.2.4 Discussion of Different Surrogate Models 61
5.3 Scalability Issues of Surrogate-assisted EAs 62
5.4 The Management of Fitness Approximation in EA 63
5.4.1 A Brief Review on Managing Surrogates 63
5.4.2 Pre-Selection Surrogate Management Strategy . . . 64
5.4.3 Best Surrogate Management Strategy 65
5.4.4 Modified Pre-Selection and Best Surrogate Management Strategy 66

6 Initial Experiments: Comparison of Different Approximation Models 71
6.1 Approximation Accuracy Measurements 71
6.2 Initial Offline Experiment Data Description 73
6.3 Initial Offline Experiment Setup 75
6.4 Initial Offline Experiment Results 77

7 Evaluation and Results 79
7.1 Wind Farm Scenarios 79

7.2 Objective Function 81
7.3 Evaluation Setup 83
7.4 Evaluation Results 84
7.4.1 Wind Farm Scenario Kusiak & Song 1 [54] 85
7.4.2 Wind Farm Scenario Kusiak & Song 2 [54] 93
7.4.3 Wind Farm Scenario 2014 Comp 1 [105] 101
7.4.4 Wind Farm Scenario 2014 Comp 3 [105] 109
7.5 Summary of Evaluation Results 117

8 Conclusions 119

Bibliography 121

List of Figures

2.1 Aerial View of Lillgrund Windfarm [91]. The Lillgrund Windfarm is located off the coast of Copenhagen in Denmark. . . 6

2.2 Layout of the Middelgrunden offshore wind farm [28]. (a) actual layout, (b) symmetrically optimised layout, (c) randomly optimised. 7

2.3 Simple drawing the rotor and blades of a wind turbine from a front point of view (left) and a side point of view (right), respectively [36]. 8

2.4 Power curve (black line) and thrust coefficient curve (grey line) of the turbine Vestas V63 [89, 98]. 9

2.5 Example of the distance between turbines in both prevailing wind direction and the direction perpendicular to the prevailing wind. There are five turbines each column and the prevailing wind blows from left to right. 10

2.6 An example of a wind rose diagram, showing statistics of wind speed and direction throughout the year [4]. This is a wind rose with 16 sectors whereas the center of each sector indicates at a given location (i.e a wind farm site). The length of each colour-coded "spoke" around the circle illustrates the relative wind speed in the pointed direction. 12

2.7 Wind farm boundary and the definition of the wind speed direction [54]. The two arrows indicate that the wind might come from both both directions. The length of each colour-coded "spoke" around the circle illustrates the relative wind speed in the pointed direction. 14

2.8 Schemetic representation of the wake effect. 15

2.9 Schemetic representation of multiple wake affecting a position. 15

3.1 Example of Crossover and Mutate Operations. 35
3.2 A cube in space formed by two three-dimention vectors (black circles). The space inside the cube represents all possible results produced by crossovering the two vectors. The outer space represents the possible results by mutating the two vectors. 37

4.1 Illustration of the BlockCopy Mutation Operation. In this example, block B_2 is chosen for mutation and copied to the position occupied by B_6. 45
4.2 Illustration of the BlockCopy Crossover Operation. In this example, block B_2 from parent B is chosen is chosen to be replaced by block A_2 from parent A. 46

5.1 Tradeoffs between computational cost and accuracy among different levels of fitness evaluations [9]. 50
5.2 Building Surrogate Models via Offline experiments. 52
5.3 One node of MLP: an artificial neuron. 58
5.4 Architecture of a multilayer perceptron network. 59
5.5 Pre-selection Surrogate Management Strategy [46]. 65
5.6 Best Surrogate Management Strategy [46]. 66
5.7 Collect and prepare wind farm layout data. The layout data consists of raw Cartessian coordinates (x_i, Y_i) of the wind turbines. Then they are converted into polar coordinates (d_i, θ_i). At the end the polar coordinates are sorted according to the distances d between the turbines and the zero point. 68
5.8 Surrogate Assisted (μ, λ) - ES Using Pre-selection Strategy Strategy with Surrogate-Retraining. 69
5.9 Surrogate Assisted (μ, λ) - ES Using Best Strategy Strategy with Surrogate-Retraining. 70

7.1 Obstacles in scenario Comp 1 and Comp 3. Layouts are not shown to scale. 80

7.2 Wind rose used in each scenario. These wind rose diagrams are not shown to scale. 81

7.3 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario. 87

7.4 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario. 88

7.5 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario. 89

7.6 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario. 90

7.7 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario. 91

7.8 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario. 92

7.9 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario. 95

7.10 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario. 96

7.11 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario. 97

7.12 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario. 98
7.13 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario. 99
7.14 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario. 100
7.15 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$. 103
7.16 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$. 104
7.17 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$. 105
7.18 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$. 106
7.19 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$. 107

7.20 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$. 108
7.21 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario. 111
7.22 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario. 112
7.23 Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario. 113
7.24 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario. 114
7.25 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario. 115
7.26 Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario. 116

List of Tables

2.1 The characteristics of a Wind Turbine. 8
2.2 Typical Surface Roughness Lengthes [28]. 11

3.1 Common terms used in evolutionary computation. 30

5.1 The dimension of optimisation problems reviewed in Section 5.2. 63

6.1 The number of turbines and number of attributes for each scenario after each filter is applied. 75
6.2 Options for the MLP classifier used in our initial offline experiments. 76
6.3 Options for the M5P classifier used in our initial offline experiments. 77
6.4 Comparison of correlation coefficient (first line in each cell) and root relative squared error (second line in each cell) on datasets for wind farm scenario Kusiak & Song 1 [54]. 78
6.5 Comparison of correlation coefficient (first line in each cell) and root relative squared error (second line in each cell) on datasets for wind farm scenario Kusiak & Song 2 [54]. 78
6.6 Comparison of (first line in each cell) and root relative squared error (second line in each cell) on datasets for wind farm scenario 2014 Comp 1 [105]. 78
6.7 Comparison of (first line in each cell) and root relative squared error (second line in each cell) on datasets for wind farm scenario 2014 Comp 3 [105]. 78

7.1 Wind Scenario dimensions, number of turbines, number of blocks and k parameter. 81
7.2 The discreption of different evolutionary algorithms. 83
7.3 The discreption of different evolutionary algorithms. 84
7.4 The average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario Kusiak & Song 1 [54]. The unit is milliseconds. 86
7.5 The average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario Kusiak & Song 2 [54]. The unit is milliseconds. 94
7.6 The average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario 2014 Wind Farm Layout Optimisation Comp 1 [105]. The unit is milliseconds. 102
7.7 The average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario 2014 Wind Farm Layout Optimisation Comp 3 [105]. The unit is milliseconds. 110

Chapter 1

Introduction

Wind power is becoming increasingly more important around the world along with the rapid development of the global economy. The Global Wind Energy Council (GWEC) reports that globally wind power production reached 433 gigawatts (GW) by the end of 2015, a cumulative 17% increase. Furthermore, GWEC predicts that by the end of 2030, there will be about 2,000 GW of wind power spinning around the world [27]. It is clear that wind power is now a mainstream source of renewable energy supply and will play a leading role in the future. The wind industry is interested in using technical innovation to drive costs down, improve project reliability and predictability, and make it easier to integrate wind power into the main power grid.

It is obvious that wind farm developers desire a high profit wind farm. In the real world, there is a trade off between two conflicting opposed economical factors: cost of construction and maintenance, and the profit of selling generated electricity. In order to reduce cost and increase profit, wind farm designers are moving towards larger sized farms, more turbines, and other advanced capabilities. However, installing significant numbers of turbines close together causes them to interfere with each other due to the wake effects. The wake effect cases the downstream wind speed to reduce which leads to a considerable loss the power production, and thus a decrease in profit.

Finding high quality wind farm layout solutions probably will significantly increase the profit for the wind farm developers. The Wind Farm Layout Optimisation Problem (WFLOP) focuses on finding the optimal turbine positions on a wind farm (wind farm layout) to gain maximum power output. Normally, the WFLOP are conveniently solved using simplified rules that lead to rectilinear layouts, where turbines are often organised in identical rows that are separated by an unnecessarily large distance [89]. Recently, a few studies have shown that irregular and nonuniform layouts (see Figure 2.2) can produce more energy than regular grid layouts [54, 88, 93].

In the literature, the related work on this topic are very limited and most of them has been carried out by the wind engineering and renewable energy communities. There are only a few works has been done by the operations researchers and meta-heuristic algorithms researchers. One reason this problem has been disregarded by the research community is the computational complexity. For example, even in the case of a simplified single objective function (maximising the wind farm energy output), one wind farm layout evaluation can take minutes, hours, or even days. Due to the high computational complexity, research on the WFLOP are also motivated to reduce the computational cost.

Furthermore, this particular problem requires the researcher to have certain specialist knowledge about how wind model (see Section 2.2.3) and wake model (see Section 2.2.4) are formulated. Additionally, it is difficult to obtain industrial data about the problem instances, which is generally kept private by the wind farm developers [89].

Population-based meta-heuristic approaches such as genetic algorithms have found successful in optimisation problems such as portfolio selection problems [84], and knapsack problems [82], among many others. However, the WFLOP is generally high-dimensional and computational expensive. Thus it is not practical to apply meta-heuristic approaches to solve the WFLOP since a relatively large number of real fitness evaluations are required to obtain a near optimal solution.

On the other hand, surrogate-assisted meta-heuristic optimisation approaches have been shown promising performance in terms of reducing computational cost in many real world expensive optimisation problems [94, 81, 1, 2, 35, 6]. The main idea is to use computationally cheap models, known as surrogates, for approximating the expensive fitness function in evolutionary approaches. However, the surrogates have not been used with the WFLOP before. There is a potential improvement by using meta-heuristic optimisation techniques in conjunction with machine learning algorithms such as decision trees, linear models and artificial neural networks. Additionally, in the literature the maximum dimension of optimisation problem solved by surrogate-assisted evolutionary approach is only 50 [64]. It is also interesting to discovery the performance of surrogate-assisted meta-heuristic optimisation approaches on four benchmark wind farm scenarios with dimensionality ranging from 200 to 1420 dimensions.

This work focuses on the WFLOP. The main research objective is to develop an approach that can obtain near optimum solutions at a reduced computational cost for practical purposes. This work evaluated several sophisticated evolutionary algorithms using surrogates along with few standard evolutionary algorithms. The choice of surrogate to use is based on a set initial offline experiments on benchmark wind farm scenarios. These experiments also discovered which features lead to the construction of an accurate surrogate model. The previously proposed highly efficient BlockCopy Mutation and Crossover operators [73] are employed to generate offsprings (see Chapter 4). In the evaluation, I compared the final results obtained by different evolutionary algorithms using different BlockCopy operators with different surrogate models and surrogate management techniques on four benchmark scenarios from the literature [54, 105].

The rest of this thesis is organised as follows. The background of the wind farm layout optimisation problem is given in Chapter 2. Chapter 3 reviews several population-based meta-heuristic algorithms from the literature and a few of their applications for the WFLOP. In Chap-

ter 5, the popular surrogate modelling techniques are briefly described, furthermore, two surrogate model management strategies are explained along with our modification to them. In Chapter 4, the efficient Block-Copy Mutate and Crossover operators previously proposed by [73] are explained. The setup and results of a set of initial offline experiment are shown in Chapter 6. The performance of the surrogate-assisted evolutionary strategy using our chosen surrogate models with the modified surrogate management strategies are evaluated on four benchmark wind farm scenarios and compared with that of the evolutionary strategy without surrogates in Chapter 7. A brief description of the scenarios and the real fitness function is also included. Finally, Chapter 8 concludes this thesis with a summary and some ideas for future work.

Chapter 2

Wind Farm Layout Optimisation Problem

The Wind Farm Layout Optimisation Problem (WFLOP) focuses on finding the optimal placement for wind turbines (WT) in a wind farm (WF). This process is referred to as micro-sitting in the literature. The micro-sitting process generally occurs after (i) the wind farm location and boundaries has been chosen, (ii) the characteristics of the site has been measured, such as the distribution of wind speeds and directions, the obstacles inside the wind farm, etc. and (iii) the model and manufacturer of the turbines (powerful turbines are usually preferred [89]) and other auxiliary infrastructure (i.e. roading networks, electrical infrastructures, etc.) have been chosen.

2.1 Wind Farm Layout Example

The optimal wind farm is considered to be one that maximises the power generation while minimising total cost. Due to the limited energy generation of individual wind turbines, a wind farm normally is constructed by installing a large number of turbines on a given terrain. Figure 2.1 depicts the aerial view of the Middelgrunden wind farm, which is located about 3.5 km off the coast of outside Copenhagen, Denmark. When it was built in 2000, it was the largest offshore farm in the world with 20

turbines and a total capacity of 40 megawatts [92]. The large and slow turning turbines of this offshore wind farm take advantage of the not strong but very consistent wind since the wind often flows briskly and smoothly over water since there are no obstructions.

Figure 2.1: Aerial View of Lillgrund Windfarm [91]. The Lillgrund Windfarm is located off the coast of Copenhagen in Denmark.

A wind farm layout represents the geographical placement of the installed wind turbines inside the wind farm. As can be observed in Figure 2.2 (a), in the Middelgrunden Windfarm, the wind turbines are placed in a regular configuration whilst maintaining a large distance between each turbine. The turbines are mainly affected by the prevailing wind direction, which is illustrated by the yearly energy rose in Figure 2.2.

Such a regular configuration is quite straightforward to design. However, some studies show that such configurations are not necessarily optimal in terms of total energy output [78, 90, 72]. Contrary to such regularly configured layouts, Figure 2.2 (b) represents a symmetrically optimised layout and Figure 2.2 (c) represents a randomly-looking but

highly optimised for the same wind farm proposed by Neubert et al. [78]. According to [78], the layouts shown in Figure 2.2 (b) and (c) can increase the annual energy output by 5% and 6%, respectively. Obviously, this improvement will significantly increase the profit of selling the electricity for the wind farm developers. From an optimisation perspective, this improvement is mainly due to the minimisation of the wake effect [28].

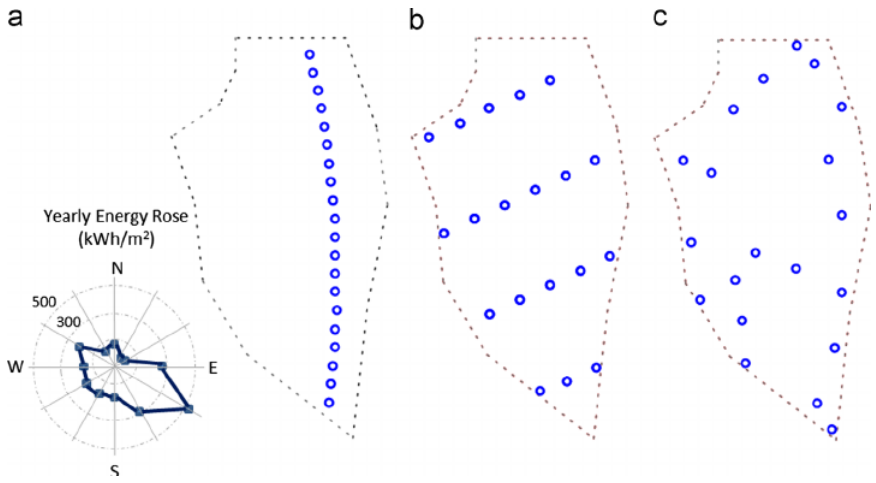

Figure 2.2: Layout of the Middelgrunden offshore wind farm [28]. (a) actual layout, (b) symmetrically optimised layout, (c) randomly optimised.

2.2 Wind Farm Wake Effect

This section give a general idea of how a wind turbine works, and how to mathematically describe the wake effect. Then it shows the findings in the literature regarding to the computational complexity of the WFLOP.

2.2.1 Wind Turbine Characteristics

The characteristics of a Wind Turbine (WT) that are related to the wind farm layout optimisation are described in Table 2.1 following:

Characteristic	Notation	Unit
Cut-in speed	c_i	m/s
Cut-out speed	c_o	m/s
Nominal speed	c_{rated}	m/s
Nominal power	P_{rated}	kW
Thrust coefficient	C_t	$0 \leq C_t \leq 1$
Rotor diamete	d	m
Hub height	z	m

Table 2.1: The characteristics of a Wind Turbine.

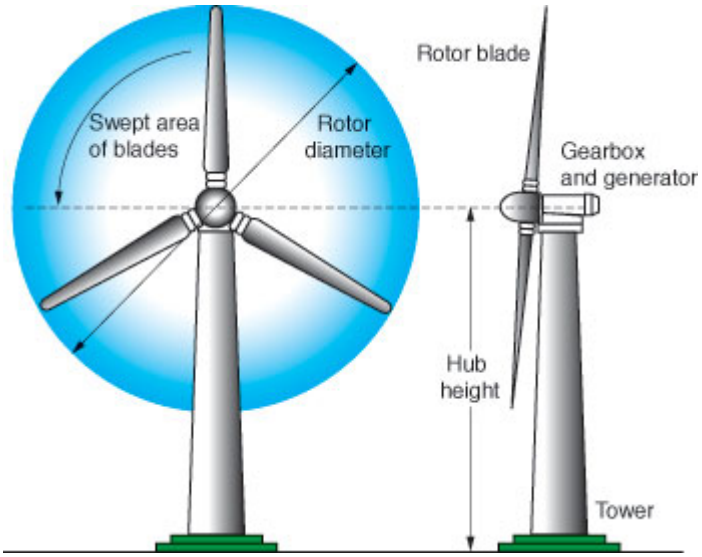

Figure 2.3: Simple drawing the rotor and blades of a wind turbine from a front point of view (left) and a side point of view (right), respectively [36].

Figure 2.3 depicts a simple drawing of the rotor and blades of a wind turbine from a front point of view and a side point of view, respectively. According to [13], in simple wind turbine designs, the turbine blades are bolted to the hub. The hub height z indicates how high the attachment point of the blades above the ground is. The hub is fixed to the rotor shaft which drives the generator through a gearbox. The rotor diameter d (also known as rotor radius) largely determines how much wind energy can be collected and turned into electrical energy. In more recent so-

phisticated designs, the blades are bolted to the pitch mechanism, which adjusts their angle of attack according to the wind speed to control their rotational speed [13].

Briefly, the turbine rotor starts to spin when the wind speed is greater than $c_i\ m/s$. The power output increases nonlinearly until the wind speed reaches the nominal speed where the turbine control system alters the pitch of the blades so that the power production becomes a constant. This is the nominal power output of the wind turbine. The power curve and thrust coefficient respectively describe the power produced and between wind speed c_i and c_o. The thrust coefficient indicates the proportion of wind energy capture when the wind passes the turbine blades. For both power curve and thrust coefficient curve, wind turbine manufacturers usually provide a few data points, which need to be interpolated to obtain the intermediate points [89]. For example, Figure 2.4 illustrates the power curve (black line) and thrust coefficient curve (grey line) of the turbine Vestas V63 ($c_i = 5\ m/s$, $c_o = 25\ m/s$, nominal speed = $16\ m/s$, nominal power = 15 Megawatts) [89, 98].

The distance between any two wind turbines is also very important since decreasing the spacing increases the turbulence induced by the wakes of neighbouring wind turbines [54, 89]. As a general rule, turbines in wind farms are usually spaced between 5 and 9 rotor diameters apart in the prevailing wind direction, and between 3 and 5 rotor diameters in the direction perpendicular to the prevailing wind [89]. Figure 2.5 shows an example the distances between turbines inside a wind farm with symmetrical constraints.

2.2.2 Surface Roughness

In a given terrain, wind speed decreases by interaction with obstacles within the terrain. For example, water surfaces are smoother than forests, and will have less influence on the wind, while long grass and shrubs and bushes will slow the wind down considerably. In short, the smoother the

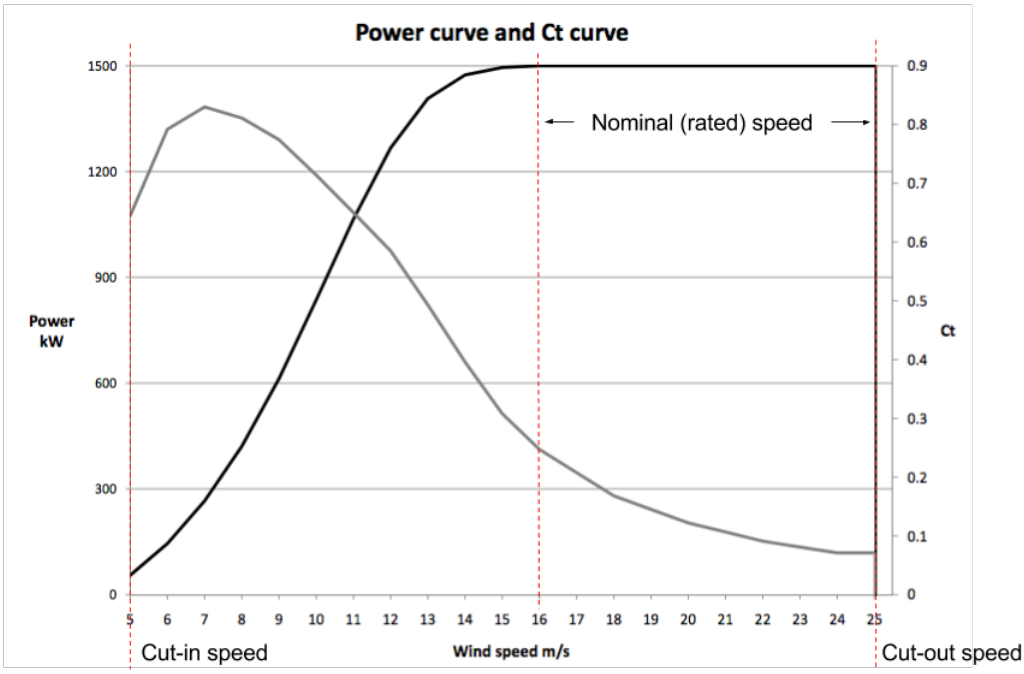

Figure 2.4: Power curve (black line) and thrust coefficient curve (grey line) of the turbine Vestas V63 [89, 98].

terrain, the less interference it has on wind. This can be modelled using a constant called roughness length z_0. Table 2.2 shows the roughness lengths of typical surfaces.

Type of terrain	Roughness length, z_0 (m)
Water Surface	0.0002
Open farmland, few trees and buildings	0.003
Villages, country with trees and hedges	0.1
Cities, forests	0.7

Table 2.2: Typical Surface Roughness Lengthes [28].

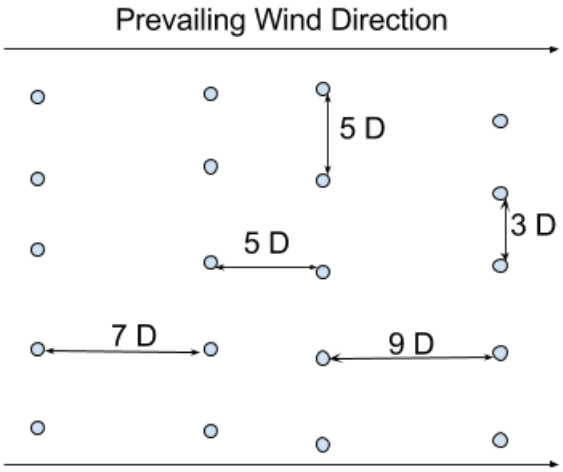

Figure 2.5: Example of the distance between turbines in both prevailing wind direction and the direction perpendicular to the prevailing wind. There are five turbines each column and the prevailing wind blows from left to right.

2.2.3 Wind Modelling

In the literature, the statistical behaviour of the wind is typically modelled by taking into account two different factors: wind direction and wind speed. A commonly used wind model is proposed by Kusiak and Song [54]. In their approach, the wind direction is represented by the probability of occurrence for each of the sectors inside a wind rose. Figure 2.6 illustrates an example of wind rose with 16 sectors. In work [75, 80, 23], the discretised distribution wind model was used with a wind rose of 36 sectors. There are other studies in the literature that have divided the wind rose into 24 sectors [93], 16 sectors [23] and 8 sectors [77, 54].

According to Kusiak and Song [54], the turbine output P is defined as:

$$P_{WT} = f(v) \tag{2.1}$$

where P is the turbine output and v is the wind speed at the turbine hub with a fixed height (see Figure 2.3).

As can be seen in Figure 2.4, a power curve $P = f(v)$ usually resem-

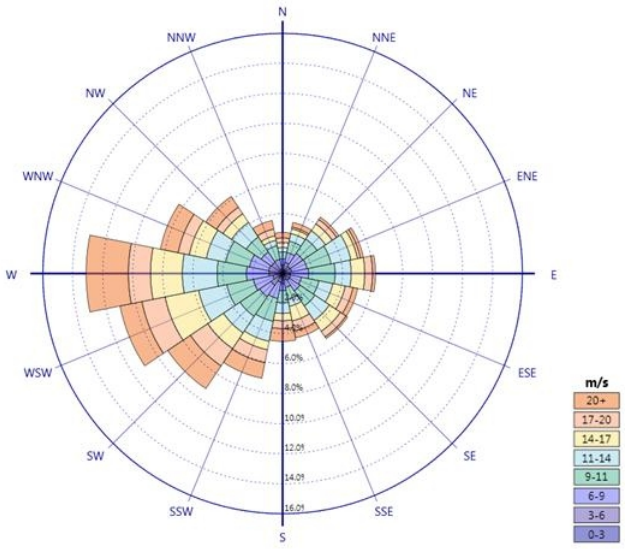

Figure 2.6: An example of a wind rose diagram, showing statistics of wind speed and direction throughout the year [4]. This is a wind rose with 16 sectors whereas the center of each sector indicates at a given location (i.e a wind farm site). The length of each colour-coded "spoke" around the circle illustrates the relative wind speed in the pointed direction.

bles a sigmoid function. It could be described as a linear function with a tolerable error. For example, in the study reported by Kusiak and Song [54], the turbine output P Equation (2.1) is expressed as:

$$\begin{aligned} P_{WT} &= f(v) \\ &= \begin{cases} 0, \ v < c_i \\ \lambda v + \eta, \ c_i \leq v \leq c_{rated} \\ P_{rated}, \ c_o > v > c_{rated} \\ 0, \ v \geq c_o \end{cases} \end{aligned} \tag{2.2}$$

where c_i is the Cut-in speed, c_o is the Cut-out speed, c_{rated} is the Nominal speed, P_{rated} is the Nominal power, λ is the slope parameter, and η is the intercept parameter. More specific, if the wind speed$v$ is smaller than Cut-in speed, there is not sufficient torque to turn the turbine and the generator, thus there is no power output. Similarly, if the wind speed exceeds the Cut-out speed, the turbine shuts down (zero output) to prevent

turbine damage. When the wind speed is between the Cut-in and the rated speed (c_{rated}), the power output can be described as a linear equation. If the wind speed is greater than the rated speed (c_{rated}) but smaller than the Cut-out, the wind turbine control system will keep the turbine rotation speed stable (fixed power output at P_{rated}) to protect the system from overloading. Once the wind speed v is greater than the Cut-out speed, the wind turbine is shut down for safety purpose and it generates zero energy.

In their approach, according to industrial wind farm data [69], the wind speed v at a given location, height and direction follows a Weibull distribution [102, 95], which can be expressed as:

$$p_v(v, k, c) = \frac{k}{c}\left(\frac{v}{c}\right)^{k-1} e^{-(\frac{v}{c})^k} \tag{2.3}$$

where $p_v(.)$ is the probability density function, k is the shape parameter and c is the scale parameter. At a given heigh, the expected wind speed v is a continuous function of the wind direction θ, because k and c can be parameterised by θ (i.e. $k = k(\theta)$, $k = k(\theta)$, $0° \leq \theta \leq 360°$). In plain english, wind speeds at different locations across the wind farm share the same Weibull distribution [54]. Figure 2.7 illustrates an example of wind roses diagram with their corresponding wind directions (i.e. $45°$ and $225°$), where East is defined as $0°$ and North is defined as $90°$ at a given location (i.e. a wind farm site).

The notation in Equations (2.1) to (2.3) is equivalent to the ones in [54].

2.2.4 Wake Effect Modelling

As shown in Figure 2.8, the wind wake effect is a phenomena occurring when a wind turbine rotor extracts a certain amount of kinetic energy from the wind flow and the downstream wind speed is reduced thus downstream turbines extract less energy [89]. Briefly, Figure 2.8 shows the Jensen wake model [44, 25]. The wind blows from left to right at

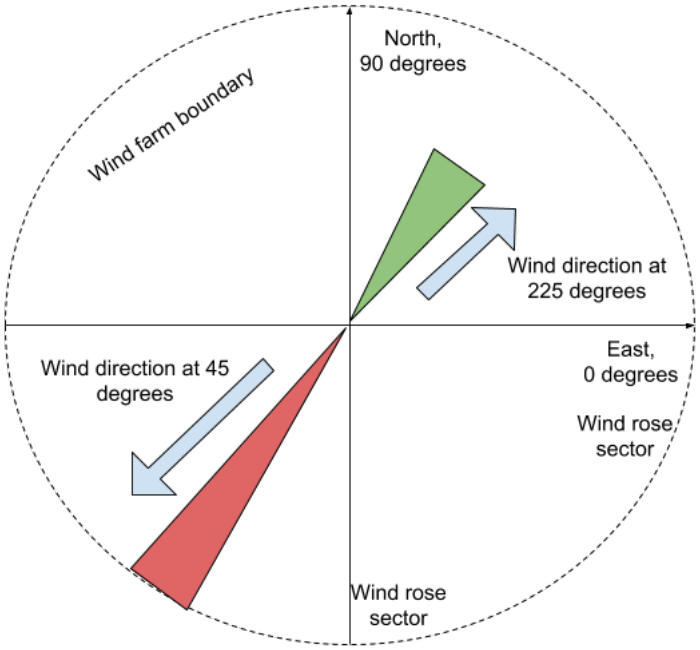

Figure 2.7: Wind farm boundary and the definition of the wind speed direction [54]. The two arrows indicate that the wind might come from both both directions. The length of each colour-coded "spoke" around the circle illustrates the relative wind speed in the pointed direction.

wind speed U_0. Then the wind hits a turbine which is represented as a black rectangle on the left. At a distance x from the turbine, the radius of the turbulence is r_1 whereas the rotor radius is r_r (equals to the initial size of the turbulence). The α is a scalar parameter that determines how quickly the wake expands with distance. Under the wake effect, the downstream wind speed reduced to $U < U_0$ whereas in the non-wake area, the downstream wind speed is still U_0. Additionally, turbulence and shear stress will increase wind load fluctuation and cause damage for those downstream turbines.

According to Jensen [44], the wind speed deficit (see Figure 2.8) caused by the airflow passes through the wind turbine rotor and it is

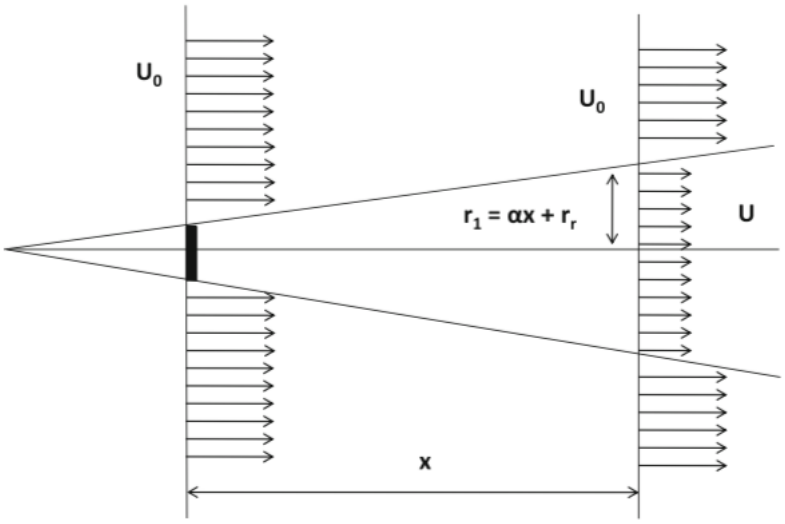

Figure 2.8: Schemetic representation of the wake effect.

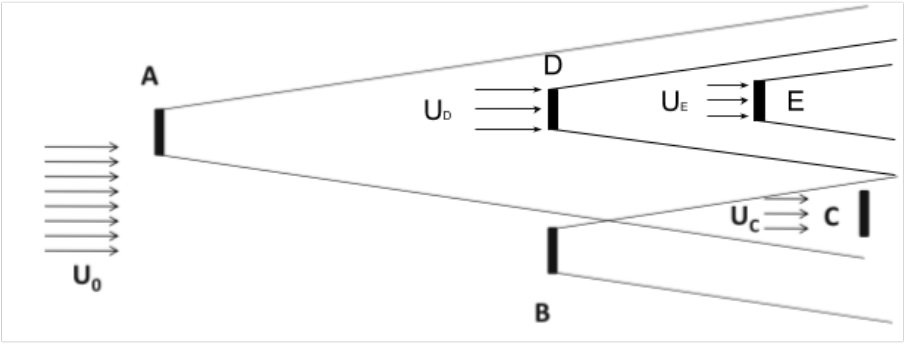

Figure 2.9: Schemetic representation of multiple wake affecting a position.

calculated as follows:

$$U = U_0 \left[1 - \frac{2a}{1 + \alpha(\frac{d}{r_1})^2} \right] \tag{2.4}$$

where d is the distance between the two turbines.

The scalar α determines how quickly the wake expands with distance

and it is can be computed by the expression:

$$\alpha = \frac{0.5}{\ln \frac{z}{z_0}} \tag{2.5}$$

where z is the hub height of the wind turbine whereas z_0 is a the surface roughness.

The term r_1 represents the radius of the wake downstream and it can be calculated by the expression:

$$r_1 = r_r \sqrt{\frac{1-a}{1-2a}} \tag{2.6}$$

where r_r is the turbine rotor radius at upstream while the term a is the axial induction factor and it is defined as:

$$a = 0.5\Big(1 - \sqrt{1 - C_t}\Big) \tag{2.7}$$

where C_t is the thrust coefficient of the turbine.

For any two turbines that are located at i and j inside a wind farm, the wind speed velocity deficit at turbine j in the wake of turbine i is defined as:

$$vel\_def_{ij} = \frac{2a}{1 + \alpha(\frac{x_{ij}}{r_j})^2} \tag{2.8}$$

where term a is defined in Equation (2.5), x_{ij} is the distance between the two turbines and r_j is the radius of the wake downstream at turbine j which is calculated by Equation (2.6). Since many turbines are installed in a wind farm, wakes can intersect with each other and affect turbines downstream at the same time. When a turbine j is affected by the wakes of multiple turbines, the total velocity deficit is computed as:

$$vel\_def_j = \sqrt{\sum_{i \in W(j)} vel\_def_{ij}^2} \tag{2.9}$$

where $W(j)$ is the set of turbines affecting the turbine located at position j.

The interaction of multiple wakes is not fully understood and is subject of many studies in the aerodynamics field [89]. Herbert-Acero et al. [37] reported that the wake effect between turbines must be calculated sequentially. As can be seen in Figure 2.9, turbine C is in the wake generated by both turbine A and B, whereas turbine B is not affected by the wake of any other turbine ($U_C < U_0$). Turbine A is placed at the upstream of turbine D and E while turbine E is located at the downstream of turbine D ($U_E < U_D < U_0$). In order to calculate the wake effect on turbine C, the wake generated by turbine A and B should both be taken into account. Similarly, the wake effect on turbine E only can be calculated until the wake effect generated by turbine A at turbine D is known.

There is another wake model proposed by Katic et al. [51]. It takes the theory of Betz [13] and the balance of momentums into consideration. The speed deficit caused by the wake is calculated as:

$$U(d) = U_0\left(1 - \left(1 - \sqrt{1 - C_t}\right)\left(\frac{D}{D(d)}\right)^2\right) \tag{2.10}$$

where D is the rotor diameter and $D(d)$ is the diameter of the downstream wake at distance d which calculated as:

$$D_W(d) = D + 2k_W d \tag{2.11}$$

where k_W is the wake effect constant. The recommended value for k_W in onshore wind farms is 0.075 whereas it is 0.05 for offshore wind farms. The notation in Equations (2.4) to (2.11) is equivalent to the ones in [89, 28].

The computational effort required for Jensen [44] wake model is similar to Katic [51] wake model. The Katic can obtain accurate results only using relatively simple mathematical formulation, thus it is the most widely used wake model in the wind energy industry [28].

2.3 Wind Turbine Energy Output

For a single turbine at location (x, y) with wind blowing from direction θ, the expected energy production is defined as:

$$\begin{aligned} E_{WT}(\theta) &= \int_0^\infty f(v) p_v(v, k(\theta), c(\theta)) \mathrm{d}v \\ &= \int_0^\infty f(v) \frac{k(\theta)}{c(\theta)} \left(\frac{v}{c(\theta)} \right)^{k(\theta)-1} e^{-(\frac{v}{c(\theta)})^{k(\theta)}} \mathrm{d}v \end{aligned} \tag{2.12}$$

where, as mentioned, $f(v)$ is the power curve and p_v is the distribution over wind speeds.

The expected wind turbine energy production for θ in the range $0 - 360^\circ$ is calculated as:

$$\begin{aligned} E_{WT} &= \int_0^{360} p_\theta(\theta) E(P, \theta) \mathrm{d}v \\ &= \int_0^{360} p_\theta(\theta) \mathrm{d}\theta \int_0^\infty f(v) \frac{k(\theta)}{c(\theta)} \left(\frac{v}{c(\theta)} \right)^{k(\theta)-1} e^{-(\frac{v}{c(\theta)})^{k(\theta)}} \mathrm{d}v \end{aligned} \tag{2.13}$$

In order to fit real data into this wind model, the power curve should be provided by the turbine manufacture whereas the statistical wind speed and direction data should be measured over a period of time on site. In order to perform numerical integration, the wind speed and its direction are discretised. It needs to be noted that continuous wind characteristics are not available in the design of wind farms [54].

Assume that the wind direction is discretised into $N_\theta + 1$ sectors (see Figure 2.6) of equal length. The dividing points of wind direction are $\theta_1, \theta_2, \ldots, \theta_{N_\theta}$ where $0^\circ < \theta_1 \leq \theta_2 \leq \cdots \leq \theta_{N_\theta} < 360^\circ$, $\theta_0 = 0^\circ$, $\theta_{N_\theta+1} = 360^\circ$. Each sector is associated with a relative expected wind speed $0 \leq \omega \leq 1,\ i = 1, \ldots,\ N_\theta$. For example, ω_0 is the expected wind speed of sector $[0^\circ, \theta_1]$ whereas ω_{N_θ} is the expected wind speed of sector $[\theta_{N_\theta}, 360^\circ]$. The

expected wind speed ω_i is estimated using the wind farm data [54].

Similarly, the wind speed data is also discretised into $N_v + 1$ sectors. The dividing points of wind direction are $v_1, v_2, \ldots, v_{N_v}$ where $v_{cut-in} < \theta_1 \leq \theta_2 \leq \cdots \leq v_{N_v} < v_{rated}$, $v_0 = v_{cut-in}$, $v_{N_v+1} = v_{rated}$.

Therefore, the wind turbine energy output can be calculated as:

$$
\begin{aligned}
E(WT)_i = \lambda \sum_{j=1}^{N_v+1} \left(\frac{\theta_j + v_j}{2} \right) \sum_{l=1}^{N_\theta+1} \Bigg\{ & (\theta_l - \theta_{l-1})\omega_{l-1} \\
& \left\{ e^{-\left(v_{j-1}/c_i\left(\frac{\theta_l+\theta_{l-1}}{2}\right)\right)^{k\left(\frac{\theta_l+\theta_{l-1}}{2}\right)}} - e^{-\left(v_j/c_i\left(\frac{\theta_l+\theta_{l-1}}{2}\right)\right)^{k\left(\frac{\theta_l+\theta_{l-1}}{2}\right)}} \right\} \Bigg\} \\
& + P_{rated} \sum_{l=1}^{N_\theta+1} \left\{ (\theta_l - \theta_{l-1})\omega_{l-1} e^{-\left(v_{rated}/c_i\left(\frac{\theta_l+\theta_{l-1}}{2}\right)\right)^{k\left(\frac{\theta_l+\theta_{l-1}}{2}\right)}} \right\} \\
& + \eta \sum_{j=1}^{N_\theta+1} \Bigg\{ (\theta_l - \theta_{l-1})\omega_{l-1} \Bigg(e^{-\left(v_{cut-in}/c_i\left(\frac{\theta_l+\theta_{l-1}}{2}\right)\right)^{k\left(\frac{\theta_l+\theta_{l-1}}{2}\right)}} \\
& - e^{-\left(v_{rated}/c_i\left(\frac{\theta_l+\theta_{l-1}}{2}\right)\right)^{k\left(\frac{\theta_l+\theta_{l-1}}{2}\right)}} \Bigg) \Bigg\}
\end{aligned}
\tag{2.14}
$$

where λ is the slope parameter, η is the intercept parameter, P_{rated} is the nominal power. The notation in Equations (2.12) to (2.14) is equivalent to the ones in [54].

Discretisation of wind direction and wind speed is commonly used in the literature. In work [75, 80, 23], the discretised distribution wind model was used with a wind rose of 36 sectors. There are other studies in the literature that have divided the wind rose into 24 sectors [93], 16 sectors [23] and 8 sectors [77, 54]. In this thesis, the directions are discretised into 24 sectors (see Section 7.1).

2.4 WFLOP Objective Functions

In the existing literature, the commonly used objective function is proposed by Kusiak and Song [54]. In their approach, all turbines are as-

sumed identical. The total number of turbines inside the wind farm is fixed and each turbine varies only in its (x, y) Cartesian coordinates on the plane. The objective is to maximise the annual total power output production of the wind farm (E_{WF}), which is obtained by summing up the contribution of all the turbines:

$$E_{WF} = \sum_{i=1}^{N_{WT}} E(WT)_i \tag{2.15}$$

where $E(WT)_i$ is defined in Equation (2.14) and N_{WT} is the number of turbines installed in the wind farm. There are two constraints for this objective function. Given the rotor radius R, any two turbines at position x_i, y_i and x_j, y_j should satisfy the inequality $(x_i - x_j)^2 + (y_i - y_j)^2 \leq 64R^2$. In other words, the minimum distance between any two turbines is $8R$. This is very important since the wake effect is considerably stronger when two turbines are too closely placed [54].

Mosetti et al. [75] proposed another approach to the problem. The maximum annual energy output (E_{WF}) is achieved with the minimum total wind farm cost ($cost_{tot}$). The objective function to minimise is defined as:

$$Obj = \frac{1}{E_{WF}} \times w_1 + \frac{cost_{tot}}{E_{WF}} \times w_2 \tag{2.16}$$

where w_1 and w_2 are the random weights and $cost_{tot}$ is defined as:

$$cost_{tot} = N_{WT} \times \left(\frac{2}{3} + \frac{1}{3} \times e^{-0.00174 \times N_{WT}^2}\right) \tag{2.17}$$

where N_{WT} is the number of wind turbines.

The maximisation of profit is studied by Ozturk and Norman [80]. The profit is calculates as:

$$Profit = \left[p_{kWh} - \left(\frac{cost_{tot}}{E_{WF}}\right)\right] \times E_{WF} \tag{2.18}$$

where p_{kWh} is the selling price of harvest energy per kilowatt-hours and $cost_{tot}$ is total cost of the wind farm calculated using the cost model in Equation (2.18).

The minimisation of the cost/energy is studied in work [31, 70]. The ratio is calculated as:

$$Ratio = \frac{cost_{tot}}{E_{WF}} \tag{2.19}$$

where the $cost_{tot}$ is calculated using Equation (2.18).

Mora et al. [74] proposed a new approach to the problem by maximising the net present value (NPV). This complex approach takes a complete economic model into account for the wind farm. The objective function to maximise is defined as:

$$NPV(\chi) = \frac{CF_1(\chi)}{1+r} + \frac{CF_2(\chi)}{(1+r)^2} + \cdots + \frac{CF_i(\chi)}{(1+r)^L T} - I_{WF}(\chi) \tag{2.20}$$

where CF_1 is the cash flow of each year, χ is the wind farm configuration, I_{WF} is the initial investment of the wind farm, r is the discount rate of money, and LT is the life time of wind farm.

Lackner and Elkinton [58] proposed an objective function for the design of offshore wind farms. It takes a number of aspects into account such as turbine cost, roading cost and electric interconnection cost. By minimising the levelized production cost (LPC), the objective function is defined as:

$$LPC = \frac{I_{WF}}{a_f E_{WF}} + \frac{C_{O\&M}}{E_{WF}} \tag{2.21}$$

where a_f is the annuity factor and $C_{O\&M}$ represents the cost of operation and maintenance. The notation in Equations (2.15) to (2.21) is equivalent to the ones in [28].

As can be seen, there are several objective functions for WFLOP in the literature. Most of the studies have used the simplified option in Equation (2.16) because it is accurate enough to show the ability of the proposed optimisation methods. However, the complex objective functions are more realistic as they take the economical behaviour of the project into account. They can be used to analyse the influence of aspects such as roading cost, electric infrastructure cost and other auxiliary costs.

Our work employ an extended version from 2015 Wind Farm Layout Optimisation Competition[105], which was originally proposed by Kusiak and Song [54]. By calculating the total cost of the farm (including construction and yearly operating costs) and dividing that by the total power output of the wind farm, the cost is defined as the expected cost of per kilowatt energy output. The objective function to maximise is defined as:

$$cost = \frac{(c_t \times n + c_s \times \lfloor \frac{n}{m} \rfloor)(\frac{2}{3} + \frac{1}{3} \times e^{-0.00174n^2}) + C_{O\&M} \times n}{(1-(1+r)^{-y}) \Big/ n} \times \frac{1}{8760 \times P} + \frac{0.1}{n} \tag{2.22}$$

where $c_t = 750,000$ is the cost of a turbine in USD; $c_s = 8,000,000$ is the cost of subsection in USD; $m = 30$ is the number of turbines per subsection; $r = 0.3$ is the interest rate; $y = 20$ is the lifetime of the farm in years; $C_{O\&M} = 20,000$ is the cost of operations and maintenance in USD; n is the number of turbines; and P is the total energy output of the farm.

This objective function is based on the Jensen wake model [44]. It is robust [26], thus it can provide adequate accuracy for wind farm simulations. Mayo and Zheng [73] previously also used this objective function to evaluate the performance of a novel evolutionary search operator for the WFLOP. This objective function is simplified, for example, the runtime (3.4 GHz Intel Core i5) for one evaluation using this objective function on a wind farm scenario with 100 WTs and 720 WTs are approximately 410 milliseconds and 8200 milliseconds, respectively. Despite the simplification, it is still adequately accurate [26].

2.5 Wind Farm Design Tools

According to a recent review paper done by González et al. [28], there are several commercial available software packages that help wind farm designer assess their designs. The most popular one is WAsP [18], which is designed specifically for wind resource assessment and siting of wind turbines and wind farms in various terrain. A module of WAsP uses the

Katic wake model [74] to calculate the energy output of the wind farm by taking into account extreme wind conditions, wind sheers and turbulence, etc. The recent update added the WAsP CFD module, which uses a computational fluid dynamics (CFD) wind model that allows wind farm designer to test their designs on complex terrain. WAsP CFD includes an useful online calculation service where high quality WAsP CFD calculations are performed on a high-performance computer cluster via the internet.

Similarly, WindSim [106] offers the wind farm design assessment functionality using a CFD model based on a 3D Reynolds-averaged Navier-Stokes solver. This tool can identify the spots within the wind farm that have better wind speed condition and low turbulences, thus the designers can place the wind turbines on more potentially suitable locations.

Both the WAsP and WindSim software are powerful in terms of assisting wind farm designers to make wind turbine micro-sitting decisions, however, both the WAsP and WindSim only focus on the assessment of the wind farm design based on annual energy output. The problem of optimising the layout of wind farms is not the main purpose of these tools.

There are few software packages that tackle the wind farm layout optimisation problem. Windfarmer [24] optimises the layout of a wind farm in order to maximise the return of investment. But the developers provide no information regarding to the optimisation method or the objective function used. Thus performance in terms of accuracy is a concern.

Another package called WindPro [20] focuses on finding better wind farm layouts by maximising the annual energy output using the Katic wake model [74]. In particular, the software incrementally adds turbine to candidate layouts then evaluates. It can handle random turbine configurations as well as symmetrical turbine configurations (i.e. see Figure 2.2).

OpenWind [5] is an open source software which minimises the cost of energy production using the deep-array wake model [11], which is

based on the Katic wake model [74]. But no further details are provided regarding to the optimisation method.

As can be seen, although there are some commercial and open source software packages are available, they are mainly designed for evaluating potential wind farm layouts rather than optimising them. The details regarding to the optimisation algorithms employed are very limited.

In this thesis, we used a simulator from 2015 Wind Farm Layout Optimisation Competition[105]. The simulator employs a grid-based genetic algorithm for search near optimal solutions. The objective is to minimise the cost of energy. It uses an extended version of the objective function proposed in work [54]. The objective function is shown in Equation (2.22). The simplicity of this simulator makes it very efficient to run. The details regarding to the employed optimisation algorithm and objective function are very helpful for our study.

2.6 Variants of the WFLOP

In the literature, several variants of the WFLOP have been investigated. The number of turbines is one of the most significant issues in the design of a new wind farm. In study [22], the proposed approach can be useful for the estimation of the optimal number of wind turbines in a wind farm. The wind farm initial cost can be significantly reduced by only using the minimum required number of wind turbines for specific power productivity.

The power quality issue is also a concern for wind farm designers. From the customer's point of view, it is desirable to have electricity without fluctuating voltage and frequency at the receiving end. This issue can be addressed at the wind turbine or the wind farm level. In work [76], the potential power quality improvement of a single wind turbine was analysed in conjunction with a diesel generator. Later on, the modelling and control techniques for such wind-hybrid power generation systems

were studied in [53] to enhance the power quality on a given wind farm.

In another version of the WFLOP, the landowners are taken into the consideration of wind farm design in work [14] and [100]. The authors pointed out that currently research on WFLOP assumes a continuous piece of land is readily available and focuses on advancing optimisation methods, however, in the real world projects rely on landowners' permission for success. The landowners should be consulted in order to find the most cost-effective plots of land to construct the wind farm, therefore the total cost of the wind farm can be reduced.

In yet another version of the WFLOP, the auxiliary infrastructures such as electrical infrastructures and roading networks are studied in work [3, 29]. They pointed out that the auxiliary costs should be kept in mind to calculate the initial investment, so that designers can accurately optimise the wind farm. These added variables lead to a problem that there is no analytic function to model the wind farm costs. This fact makes the problem non-derivable, preventing the use of classical analytical optimisation techniques [29].

There are few studies consider the environmental impacts of wind farms. For example, study [57] have presented a continuous-location model for layout optimisation that take noise propagation and energy generation as objective functions. Similarly, work [78] proposed a method generating visually appealing symmetric layouts that preserve cultivated geometric regularities, without compromising the energy output.

Despite these variants, most of the studies in the literature (as well as this work) focus on the two dimensional planner version of the WFLOP [54], in which all turbines are assumed identical. The number of turbines n inside the wind farm is fixed and each turbine varies only in its (x, y) Cartesian coordinates on the plane. Thus a wind farm with N turbines is represented as: $\{(x_i, y_i), i = 1, \ldots, N\}$. The wind turbines are assumed to be identical so that they all have the same hub height and energy output. This simplified variant of the WFLOP is the most widely tackled version. Although simplified, the WFLOP is still a high-dimensional and computa-

tionally expensive optimisation problem. For example, in this thesis the four benchmark wind farm scenarios (see Section 7.1) are formed from 100 to 720 wind turbines. Thus the dimensionality ranges from 200 to 1420.

2.7 Computational Complexity of WFLOP

Currently, the primary method for evaluating wind farm layout is via CFD simulation. The problem is the tremendous computational complexity. For example, work [33] reported that simulating wind farms with non-flat topography can take up to 8-10 hours per simulation. Research on the WFLOP is often motivated to reduce computational cost for such expensive simulation evaluation.

In a wind farm simulation, if the wind speed and direction data is given, then the wake effect between pairs of turbines can be calculated for each wind direction. Subsequently, the overall energy production of the farm can be computed [73]. There are a few mathematical models that accurately describe the wake effect, both in terms of wind speed reduction and turbulence intensity [50, 103, 97]. But these highly complex computational fluid dynamics approaches are very computationally expensive. Some of these models are only valid for the wake that are generated far from the turbine (far wake models) and others are only valid for the turbine closely generated wake (near wake models). This is why the simplified objective function illustrated in Equation (2.22) is used.

An important point to note is that the time complexity of evaluating a wind farm layout is a polynomial function of the number of turbines. Thus the wind farm layout optimisation problem lies in the class Non-deterministic Polynomial Optimisation (NPO)-complete [37]. A common characteristic of such problems is that they do not have a known efficient solution procedure [37]. Moreover, the wake effect between turbines must be calculated in sequential order thus the evaluation function

can not be parallelised [37]. As mentioned before, in Figure 2.9, the wake effect on turbine C is subject to both turbine A and B whereas the wake effect on turbine E cannot be calculated until the wake effect on turbine A and D is known. Therefore, the interactions must be modelled individually and be computed sequentially.

In terms of the search space, WFLOP is continuous, constrained, and non-differentiable. Turbines can be placed at any valid location inside the farm. The validation of turbine placement includes the minimum distance (which is typically a distance of eight times of the radius of the turbine rotor) constraint check between turbines, and the obstacle collision constraint check. The search space can be additionally discretised, so that the original continuous optimisation is converted into one of combinatorial optimisation where a position either contains a turbine or empty [75]. In this case, the size of the search space is at least $\mathcal{O}(2^n)$ where n is the number of turbines [37].

In other words, due to the high computational complexity, even for the simplest objective (i.e. the maximisation of the wind farm energy output), the WFLOP consists of both continues and discrete variables, therefore, it cannot be completely described in an analytical form and it cannot be solved by classic optimisation methods. Consequently, most research on this particular problem utilises meta-heuristic approaches.

Chapter 3

Evolutionary Algorithms

The most common way to classify heuristic methods is based on trajectory methods vs. population-based methods [7]. More specifically, trajectory meta-heuristics use a single solution during the search process and the outcome is a single optimised solution. But due to the problem of escaping local optima, population based meta-heuristics are more commonly used. In these algorithms, a population of candidate solutions evolve during given iterations then return a population of solutions when the algorithm terminates. Population-based methods can avoid local optima issues [67].

These population-based methods are inspired by population biology, as a consequence, they also borrow the technical terms from genetics and evolution [67]. These terms are quite prevalent and generally makes sense in computer science field. The common terminology used in population-based methods are given as follows:

Common Term	Description
population	set of candidate solutions.
individual	a candidate solution.
fitness	quality of a solution.
fitness evaluation	computing the fitness of an individual which may be very computationally expensive.
child and parent	an child (offspring) is the modified copy of one or two parent solutions.
selection	picking individuals for reproduction based on their fitness.
mutation	an evolutionary operator to modify an individual.
crossover	an evolutionary operator which takes two individuals as parents, swaps sections of them, and produce one or two offsprings.
breeding	a procedure to generate one or more offsprings from a population of parents through mutation or crossover operation.
intermediate generation	a generation of individuals during the evolutionary process

Table 3.1: Common terms used in evolutionary computation.

3.1 Genetic Algorithms

Genetic Algorithms (GAs) were first introduced by Holland [39] at the university of Michigan in the 1970s. GA usually iterates population fitness evaluation, selection and breeding, and offspring population generation.

A GA usually begins with a population of randomly generated $popsize$ number of individuals. It then iterates as follows. First all the individuals are evaluated for fitness. The comparison procedure maintains a global fittest individual $Best$ (initially set to empty). At each iteration, the current $Best$ solution will be compared against the population of new individuals. Secondly, to breed, an new offspring population is prepared and two parents are selected from the original population. We then copy them, cross them over with one another, and mutate the results. This generates two children to be added to the offspring population. This breeding is repeated until the offspring population is fully filled to $popsize$. The iteration continuous with the offspring population. Algorithm 1 depicts GA pseudocode.

Algorithm 1 The Genetic Algorithm

```
1: popsize ← desired size of population

2: P ← { }                                   ▷ Population Initialisation
3: for popsize times do
4:     P ← P ∪ new random individuals
5: Best ← □
6: repeat
7:     for each individual P_i ∈ P do
8:         EvaluateFitness(P_i)
9:         if Best = □ or Fitness(P_i) > Fitness(Best) then
10:            Best ← P_i)
11:    Q ← {}
12:    for popsize / 2 times do
13:        Parent P_a ← SelectWithReplacement(P)
14:        Parent P_b ← SelectWithReplacement(P)
15:        Children C_a,C_b ← Crossover(Copy(P_a),Copy(P_b))
16:        Q ← Q ∪ {Mutate(Copy(C_a)), Mutate(Copy(C_b))}
17:    P ← Q
18: until Best is the ideal solution or we have run out of time
19: return Best
```

Please note these three functions $SelectWithReplacement$, $Crossover$ and $Mutate$ will be discusses later.

3.2 Evolutionary Strategies

The Evolutionary Strategies (ESes) were originally developed by Ingo Rechenberg and Hans-Paul at the Technical University of Berlin in the mid 1960s [21]. Similar to GA, ES iterates fitness evaluation, parents selection and breeding, and offspring population reassembly. However, whereas a GA usually selects a few parents and little-by-little generates offspring individuals until enough have been created, an ES usually em-

ploys a simple selection procedure and $then$ mutates the chosen parents to produce enough offspring individuals to replace the discarded parents.

The (μ, λ)-ES is the simplest version among the evolutionary algorithms. ES also starts with a randomly generated population of λ individuals, and then it is repeated as follows. The individuals are evaluated for fitness values, then the μ fittest ones are selected to be parents and the rest of the individuals are deleted. Each of the μ parents are used to produce λ/μ children via mutation. Overall there are λ new offspring to be put back into the population. In other words, the number of μ parents are selected to produce λ offspring in each iteration. Notice that λ should be a multiple of μ. Algorithm 2 illustrates (μ, λ) ES pseudocode.

Algorithm 2 The (μ, λ) Evolutionary Strategy

$\mu \leftarrow$ number of parents selected
$\lambda \leftarrow$ number of children generated by the parents

$P \leftarrow$ { } ▷ Population Initialisation
for λ `times` **do**
    $P \leftarrow P \cup$ `new random individuals`
$Best \leftarrow \square$
repeat
    for `each individual` $P_i \in P$ **do**
        `EvaluateFitness(`P_i`)`
        if $Best = \square$ or Fitness(P_i) > Fitness($Best$) **then**
            $Best \leftarrow P_i$
    ▷ Parents Selection
    $Q \leftarrow$ the μ individuals in P whose Fitness() are greatest
    $P \leftarrow$ { }
    for `each individual` $Q_j \in Q$ **do**
        for $\lambda \,/\, \mu$ `times` **do**
            $P \leftarrow P \cup \{Mutate(Copy(Q_j))\}$
until $Best$ is the ideal solution or we have run out of time
return $Best$

3.3 Evolutionary Strategies with Elitism

Elitism is a quite simple concept in population-based approaches: the fittest individual or individuals from previous generation are injected into the next generation of population. The (μ, λ)-ES with Elitism iterates similarly to a normal (μ, λ)-ES, but the primary difference is in keeping the desired number of n fittest individual(s) in the population P rather than deleting all the individuals from the previous generation. The breeding produces n less offsprings since those n elites will be put back into the new generation of population after breeding. Elitism is believed to be an important ingredient in population-based approaches, for example in work [109, 60] related to multi-objective evolutionary algorithms, experimental results showed that elitism is beneficial. Algorithm 3 depicts ES with Elitism pseudocode:

Algorithm 3 The (μ, λ) Evolutionary Strategy with Elitism

```
1: μ ← number of parents selected
2: λ ← number of children generated by the parents
3: n ← desired number of elite individuals

4: P ← { }                                          ▷ Population Initialisation
5: for λ times do
6:     P ← P ∪ new random individuals
7: Best ← □
8: repeat
9:     for each individual P_i ∈ P do
10:        EvaluateFitness(P_i)
11:        if Best = □ or Fitness(P_i) > Fitness(Best) then
12:            Best ← P_i
13:    ▷ Parents Selection
14:    Q ← the μ individuals in P whose Fitness() is greatest
15:    ▷ Elites Selection
16:    P ← {the n fittest individuals in P, breaking ties at random}
17:    for each individual Q_j ∈ Q do
18:        for λ/μ − n times do
19:            P ← P ∪ {Mutate(Copy(Q_j))}
20: until Best is the ideal solution or we have run out of time
21: return Best
```

3.4 Selection, Mutate and Crossover

In both GA and ES that we discussed above, there is a procedure known as $selection$. In the literature, $SelectWithReplacement$ is referred many selection techniques. A popular one is called Fitness-Proportionate Selection where an individual is selected more frequently if it has a higher fitness. It can also select some low fitness individuals occasionally [67]. In an ES, a popular selection procedure is known as Truncation Selection where the μ best parents are fixed and predefined because they are selected based on their fitness values.

Figure 3.1 illustrates the general idea of three classic ways for doing crossover in binary vectors: (a) One-Point crossover, (b) Two-Point crossover and (c) Uniform crossover. Assuming we have l elements in the vector, One-Point crossover swaps all the elements after a given index a (where $1 \leq a \leq l$) whereas Two-Point crossover swaps all the items in between two given indexes a and b (where $1 < a < b \leq l$). Uniform crossover swaps the elements located at individual indexes. Figure 3.1 (d) gives the general idea of doing mutation in a vector. The Mutate operator simply changes the elements at given indexes.

Luke [67] point out that the problem with One-Point crossover. If certain patterns (i.e. few elements has to be sequentially organised) are necessary in order to get high fitness, it is quite likely that the crossover would break up good patterns that the algorithm discovered. The solution is to use Two-Point crossover so the patterns can be preserved between the two chosen indexes. Another solution is to treat all the indexes fairly so that each point can be swapped independently which is Uniform crossover. In Figure 3.1, the vectors only consist of binary value. Crossover and mutate operation however can deal with floating-point values, for example, by averaging the values from two indexes instead of just swapping them (crossover) or adding random gaussian noise to values (mutation).

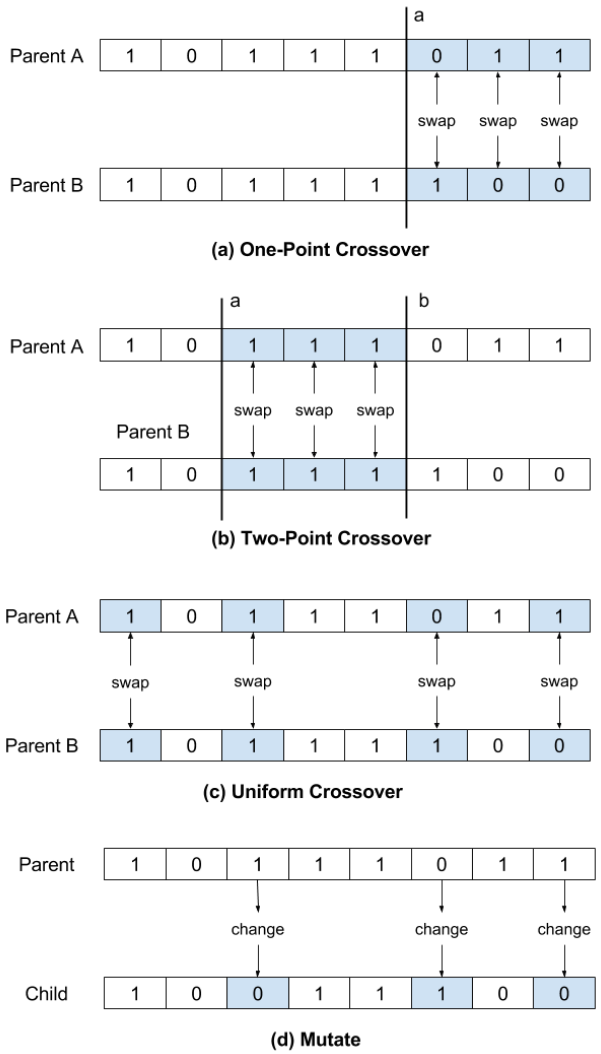

Figure 3.1: Example of Crossover and Mutate Operations.

3.5 Crossover vs. Mutation

In conventional genetic algorithms, the purpose of mutate operation is to increase diversity among the individuals within a population whereas the crossover operator aims to increase convergence. However, crossing two vectors cannot give every conceivable vector out of it [67]. As shown in Figure 3.2, the two black circles represent two three-dimensional vectors at the extreme corners of a hypercube in space. The crossovers of these two corners only results in new vectors which are located at some other

corners of this hypercube. In other words, the new vectors are still inside this hypercube. This theory can be extended to the population-based methods. Imagine the the individuals of a population P as points in the three-dimensional space in Figure 3.2. The results of crossover done on P can only be inside the bounding box surrounding P in space [67], in other words, using crossover alone is not possible to search the space outside the bounding box of P. Thus crossover is not capable of global search alone.

To further illustrate, if crossover and selection are repeated on a population many times, the population may end up in a situation where certain values for certain indexes in the vector have been eliminated. As a result, the bounding box collapses in that dimension. In other words, the individuals are quite similar or even identical to each other within in the population. The population will pre-maturely converge. At this stage, individuals are crossed with themselves and nothing new is generated. Mutation, on the other hand, introduces conceivable new values for the vector. This is why a mutate operator is required.

Despite this, crossover is still needed. It is believed that good fitness individuals share certain features in common. In many cases for which crossover was helpful, the fitness value of a given individual is at least partly in relation to these common features [67]. As mentioned before, One-point and Two-point crossover may break up these shared features, however if the vectors are carefully constructed (i.e. putting related features next to each other), it is possible that crossover can spread the features of a parent with high fitness throughout the population.

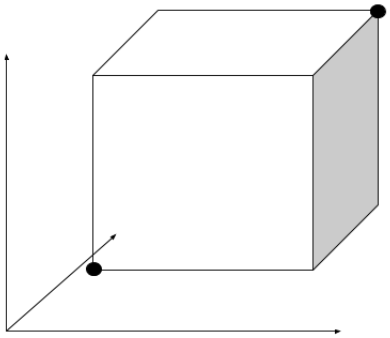

Figure 3.2: A cube in space formed by two three-dimention vectors (black circles). The space inside the cube represents all possible results produced by crossovering the two vectors. The outer space represents the possible results by mutating the two vectors.

3.6 Exploration vs. Exploitation

Generally speaking, exploration is related to global search whereas exploitation is related to local search. In a population-based optimisation context, exploitation consists of probing a limited (but promising) region of the search space with the hope of improving a high fitness individual that we already have at hand. Exploration, on the other hand, consists of searching a much larger portion of the search space with the hope of finding other promising solutions that are yet to be refined.

The search procedure is concerned with the diversification, which is the degree of difference among individuals within a population. It is clear that we need a good level of diversification in the population in order to explore the search space properly. In theory, a low level of diversification means the algorithm is close to the final solution. However, a low level of diversification in the beginning of a search leads to a premature convergence, which is far from ideal. In other words, when the diversification is high, the exploration is intensified, otherwise the local search intensifies.

In GAs and ESes, we can adjust the degree of exploration versus exploitation by tuning the population size. If the population size approaches ∞, the algorithm intensifies exploration, which ultimately leads to a random search. In the (μ, λ)-ESes, the population size is λ where as the parameter μ enables us to controls how selective the algorithm is. That is, low values of μ with respect to λ intensifies the exploitation as only the best individuals can survive. In the case of ES, with elitism, by keeping the larger number of n fittest individual(s) in the population with respect to the population size, it is more likely to search the neighbourhood space around the elites.

3.7 Evolutionary Algorithm Applications for WFLOP

This section describes the published works regarding to WFLOP. The research on find farm started over two decades ago and the number of yearly published papers addressing this problem has been increasing during the recent years [89]. Some of these works focus on wind weka model whereas some of them investigate on the wind farm layout optimisation. Due to the high computational complexity involved in WFLOP, currently, the population-based approaches are commonly used among these articles.

In the 1990s, Mosetti et al. [75] first published a work on WFLOP. In their approach, a wind farm is modelled as a 10 by 10 square grid, where the centre of each square is a possible position for a turbine. They did not predefine a fixed number of turbines to install, so their goal was to maximising the power output while minimising the installation cost (see Equation (2.16)). They used a GA based method to evolve a population of solutions through combinations and selections. A solution was represented by a vector of 100 binary variables T_i (with $i = 1, \ldots, 100$), with the presence of a turbine in position i indicated by each variable $t_i \in T_i$. New solutions are generated using crossover and mutation evolutionary oper-

ators as described in Section 3.4. They conducted a set of experimental runs using a population of 200 candidate solutions and let it evolve for 400 iterations. The computational results showed that the solutions obtained by the GA outperformed the random turbine placement method in terms of the evaluation function [75] value.

Later on in 2005, Grady et al. [31] modified the GA's parameters in the experiments presented in [75] and obtained better solutions. In essence, they divided the entire population into 20 subpopulations, then let them evolve in isolation from each other for 3000 iterations.

In 2010, González et al. [30] proposed an evolutionary algorithm to optimise wind farm layouts. The algorithm calculates the yearly income (NPV) based on energy selling price, and taking into account the investment, the wake effect, and terrain roughness. These variables add computational load to the evolutionary algorithm, but it is able to produce favourable results. The authors setup a series of algorithm performance comparisons between the proposed algorithm and the algorithm proposed by Grady et al. [31]. On one test scenario, they reported that the proposed algorithm obtained the optimum solution in 81 generations instead of the 3000 generations required by Grady's algorithm. According to them, the computational cost was drastically reduced 37.5 times. On another two test scenarios, their algorithm found a solution leading to an increase of 3.2% and 1.4% in the output energy compared to Grady's algorithm, at 14.35 times and 8.77 times less computational cost, respectively.

More recent improvement for the GA approach are reported in works [41] and [93]. Although their findings are interesting, their method only considers the case where wind blows from a constant direction at a constant speed. This simple approach cannot compute the wake effect accurately when the wind blows from more than one direction [89]. Very recently, the grid-based GA approach is used as the benchmark in the wind farm layout optimisation competition [105].

Use of ESes in WFLOP not limited to turbine placement issues, in

work [55], an evolutionary algorithm is applied to solve the data-derived optimisation model and determines optimal turbine control settings. In particular, the evolutionary algorithm was used to determine the optimal turbine blade pitch angle that can maximise the energy capture from the wind. It is worth mentioning that a multi-layer perceptron (MLP) neural network was employed to predict wind turbine energy output. The authors used the controllable variables (i.e. blade pitch angle) and non-controllable variables (i.e. wind speed) as the attributes of input nodes whereas the class value was the system response (corresponding) variables (i.e. active power). The number of hidden units was set between 5 and 20, and the weight decay for both the hidden and output layer varied from 0.0001 to 0.001. The authors reported that the MLP algorithm captured the wind turbine dynamics with highest fidelity (accuracy).

Lückehe et al. [66] presented a comprehensive investigation on a variety of different ESes for finding optimal layouts on some challenging scenarios. The objective was to maximise the wind farm energy output. They proposed a new self-adaptive $(1 + \lambda)$ evolutionary algorithm and compared it to other existing evolutionary algorithms. The proposed algorithm randomly picks n turbines in every generation then move them to new positions. How many turbines n shall be moved at the same time is controlled self-adaptively. There is also an option that the algorithm may only removes one turbine and replace it to a new position. Similarly, the probability $p \in [0, 1]$ that determines how often the algorithm moves or replaces a turbine is also operated self-adaptively. With randomly or chessboard (grid-based) initialised layouts and a population size of 50 candidate solutions, the authors reported that after about 2,000 to 5,000 fitness function evaluations, the self-adaptive $(1 + \lambda)$ started to outperform other algorithms in terms of wind farm energy output depending on scenarios and the initialisation method. It is worth mentioning that the authors presented an useful way of adjusting the exploration versus exploitation via tuning the parameters n and p. In other words, a large value of n is more like to discover a new promising design region whereas a small value of p will lead to more thorough local search.

This work utilises a standard (μ, λ) evolutionary algorithm that was presented by Luke [67]. The (μ, λ)-ES uses a population of λ randomly generated individuals. Then the iteration begins with fitness evaluation for all λ individuals. Then μ fittest solutions are chosen to produce λ/μ children through mutation and crossover operations. The low fitness parents are replaced by children. After that there is a set of new λ individuals for the next iteration of the algorithm.

Chapter 4

BlockCopy Operators for WFLOP

In this Section, the BlockCopy operators are described in more detail. The basic concept behind the BlockCopy operations is that a small part of a layout is optimised first, and then another small part of the layout is replaced by the optimised one. The wind farm is evenly divided into small square blocks, then the turbines is copied from the source block(s) to the target block. The copying means that the turbines in the target block are deleted from the wind farm layout, then the block is filled with turbines from the source block. In this way the turbine placement pattern from source block is maintained during the BlockCopy operation.

In the literature, a powerful algorithm for optimising a wind farm layout is the Turbine Displacement Algorithms (TDA) proposed by Wagner et al. [99]. The basic idea of TDA is to shift a single turbine to new random location at a time, then evaluate the modified layout. The TDA follows a simple rule that only the K nearest turbines are important for the turbine in question. The TDA tends to move the given turbine away from the K nearest turbines. The random perturbation operator is another popular evolutionary operator for the WFLOP. In short words, it randomly selects a fixed number of turbines, then moves them to new randomly selected valid locations. This approach is likely to disrupt local configurations of turbines.

Mayo and Zheng [73] evaluated the BlockCopy operators in conjunc-

tion with a $(1 + 1)$-ES across four benchmark wind farm scenarios. The TDA and the random perturbation operator are chosen for comparison. Their evaluation results of BlockCopy operators suggested that the they can be used in either a mutation or crossover context, and they are more efficient than the TDA and the random perturbation approach. Another problem about the regular crossover is that it will violate too many wind farm layout constraints (i.e. outside the layout). Therefore, in this thesis both BlockCopy Mutation operator and BlockCopy Crossover operator are used in the surrogate-assisted (μ, λ) evolutionary algorithm.

In work [73] as well as in this thesis, the size of the blocks are fixed to 1000 by 1000 metres. Thus the number of blocks can be determined during the initialisation phase. In work [73], I contributed a set of experiments to evaluate the BlockCopy operators in conjunction with a $(5, 10)$-ES on the same wind farm scenarios. Two variants of the $(5, 10)$-ES were evaluated. In the first variant, only the BlockCopy crossover operator was used where the two parent are randomly paired up (i.e. 10 individuals are paired up for 5 pairs of parent) to generate offspring. The second variant of the $(5, 10)$-ES only employed the BlockCopy crossover operator to produce offspring using two randomly selected parents.

4.1 BlockCopy Mutation Operator

The BlockCopy Mutation process is illustrated in Figure 4.1. In this example, block B_2 is chosen for mutation and copied to the position occupied by B_6. Please note that this process is performed in the same candidate layout. At first, the turbines in block B_6 are removed. Then B_6 is filled with turbines that the placement of turbines are copied from block B_2. During this operation, the relative placement configuration of the newly added turbines are maintained, and the only change is the absolute positions.

Copying turbines placement configurations across the layout may cause problems. The first problem is that the new layout might be in-

valid. For example, there is chance that if two turbines are near the edge of their respective blocks, the distance between them may be too small which may violate the minimum distance constraint (see Figure 2.5 and Equation (2.15)). To solve this problem in my implementation, the distance constraint among turbines are double checked when every single turbine is copied. If the latest added turbine violates the minimum distance constraint, it will be randomly placed in the target block to satisfy the minimum distance constraint.

Secondly, the number of turbines in the source and target block may be different. The copying in them may change the overall number of turbines of the wind farm. The fixed quantity of turbines constraint may be violated. A global correction is applied after copying in order to return the total number of turbines back to the desired fixed quantity. In particular, if the total number of turbines after copying are more than the desired fixed quantity, then some turbines will be randomly removed from the layout. Otherwise, the insufficient number of turbines will be randomly added to the wind farm to comply with the rule of global quantity.

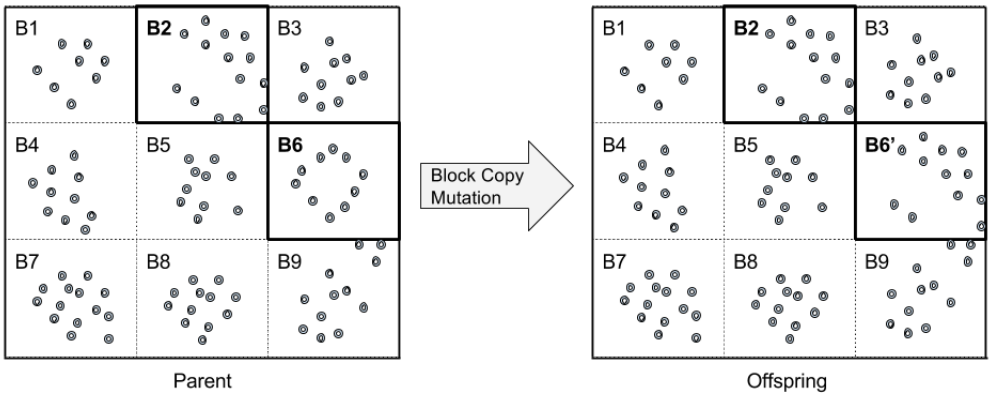

Figure 4.1: Illustration of the BlockCopy Mutation Operation. In this example, block B_2 is chosen for mutation and copied to the position occupied by B_6.

4.2 BlockCopy Crossover Operator

The BlockCopy Crossover operator performs the copying process similarly to the BlockCopy Mutation operator. The main difference is that not one but two different candidate layouts are involved: two parent layouts produce one offspring layout per crossover operation. As shown in Figure 4.2, block A_2 from parent A is selected as the source block whereas block B_6 from parent B is chosen to be the target block. The turbines inside block B_6 from parent B are deleted, then this blank block is replaced by the turbines configured in the same way as block A_2 from parent A.

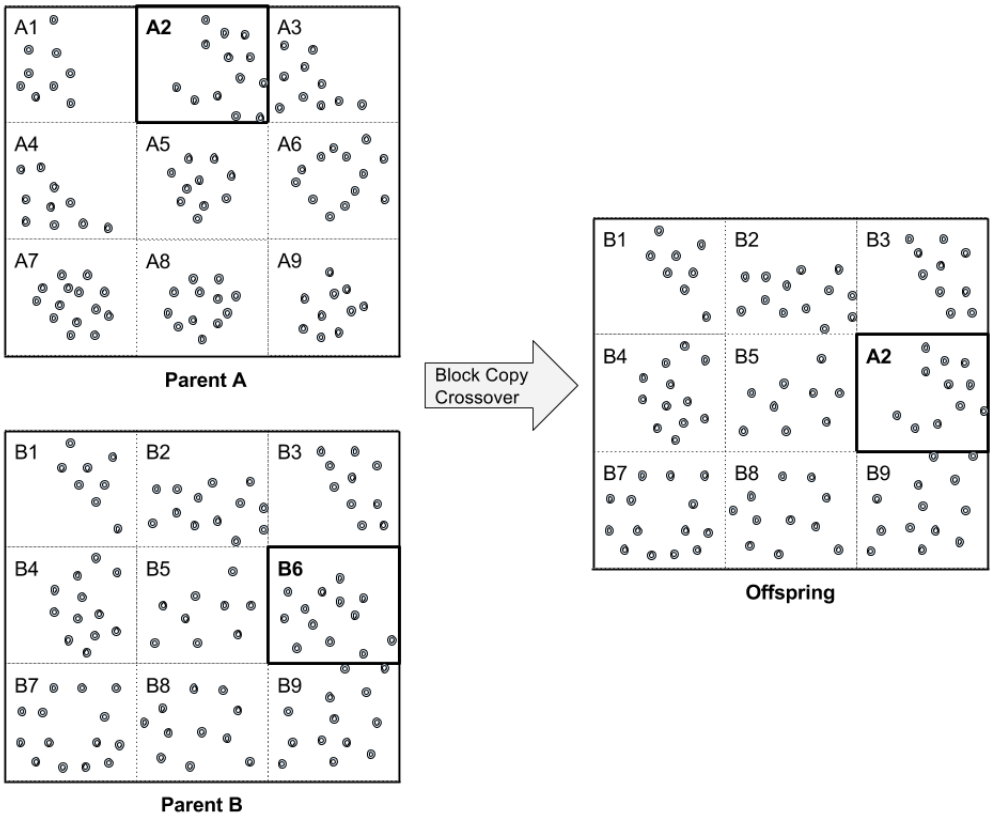

Figure 4.2: Illustration of the BlockCopy Crossover Operation. In this example, block B_2 from parent B is chosen is chosen to be replaced by block A_2 from parent A.

The problems caused by copying turbines across the layout remain in the BlockCopy Crossover operation. Figure fig. 4.2 shows an example

of solving the turbine neighbouring problem. Please note that in block A_2 from parent A as shown in , there are few turbines located in the right bottom of the block. After copying to block B_6 from the offspring, they are removed due to being too close to a few turbines at the top of block B_9 from the offspring. The overall number of turbines is maintained in the same way as the BlockCopy Mutation operation in my implementation (i.e. by randomly adding new valid turbine(s) to the layout or randomly purging invalid turbine(s) from the layout).

Chapter 5

Surrogate-Assisted Evolutionary Optimisation

5.1 Overview

As previously mentioned, evolutionary algorithms (EAs) are powerful for global optimisation. But searching for optimal solutions to complex high dimensional, multimodal real world problems generally requires expensive fitness function evaluations, such as the wind farm layout optimisation problem. Due to the computational complexity of the problem, the runtime for a single function evaluation in such challenging problems could be hours even days! The wake model and fitness function (see Equation (2.22)) used in this thesis can be thought of as an approximation to a computational fluid dynamic (CFD) model, but even that is quite expensive. For example, one optimisation using 2000 real evaluations on a wind farm layout with 720 turbines (one of the used scenarios, see Section 7.1) takes about 4.2 hours on a personal computer of Intel Core i5 CPU 3.4 GHz. Thus it is very difficult for the EAs to find satisfactory solutions in reasonable time.

An alternative is to use approximation instead of the real fitness function to reduce the computational cost [79]. Surrogate-assisted evolutionary optimisation is a class of optimisation approaches that utilises such

approximation with surrogate models to perform fitness evaluations in order to quickly find the local or global optimal solution.

Generally speaking, fitness evaluation can be obtained by experimental evaluation, complete computational simulation, simplified computational simulation, and approximation with surrogates models. The tradeoffs between computational cost and accuracy among different levels of fitness evaluations are shown in Figure 5.1. The experimental evaluation is regarded as the real fitness evaluation which can yield the real fitness value of a given candidate solution, but it incurs the highest computation cost. The complete and simplified computational simulations are less expensive as well as less accurate. Fitness evaluation by approximation with surrogate models are the cheapest to run, but also result in lowest accuracy.

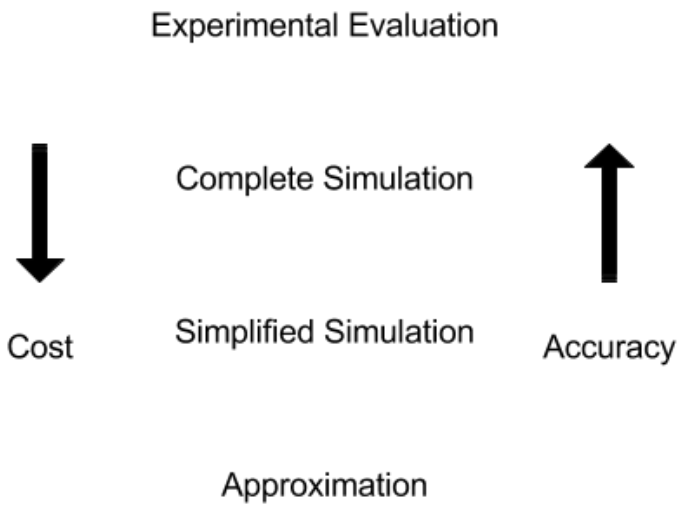

Figure 5.1: Tradeoffs between computational cost and accuracy among different levels of fitness evaluations [9].

Another advantage of surrogates models is they can smooth out the noise in the real fitness function. In a computer simulation of physical phenomena, the noise usually refers to the variation in the output due to fluctuations in the error from experiment to experiment caused by the

inputs are varied slightly [43]. For example, in a CFD simulation, the error might caused by inappropriate discretisation or incomplete convergence among other reasons. J. Forrester et al. [43] demonstrated on a set of CFD simulations that using surrogate models (in their case, the Kriging model) can approximate noisy objective function with reasonable accuracy.

Research on evolutionary optimisation using approximate fitness evaluations started about three decades ago. The first related work was reported in the mid-1980s [32], later on, a few more works were published in the late 1990s [87, 83, 12]. In 2002, the first big event devoted to research on using surrogates in evolutionary optimisation was a workshop held within the Genetic and and Evolutionary Computation Conference (GECCO) [59]. More recently, the concept of using surrogates has been adopted in more challenging real world problems such as aerodynamic design [49] and drug design [19].

5.2 Surrogate Modelling Techniques

Surrogate models are often regarded as an approximation of real fitness function(s). Surrogate modelling techniques are the methods used to build surrogate models. Normally, surrogate models are built from randomly sampled data in the design space. The accuracy of the surrogate models relies on the number of samples. It also depends on the proper selection of the approximation model to represent the real fitness function. The surrogate model will be built many times during the optimisation process, thus computational efficiency becomes a major issue of their construction process. The most widely used surrogate models, including response surface models, kriging models and artificial neural networks are discussed in this section along with their real world application(s) in the literature.

According to the literature, the basic idea for building a surrogate model is quite simple. As can be seen in Figure 5.2, the first step is to

sample data points from the design space. Then these samples are evaluated using the real fitness function. Some of them are added to the training data, the rest of them will be used for testing. The surrogate models are built based on the training data, then tested on the testing data. The sampling and building procedure should be repeated whenever the surrogate model cannot provide satisfactory performance (i.e. accuracy).

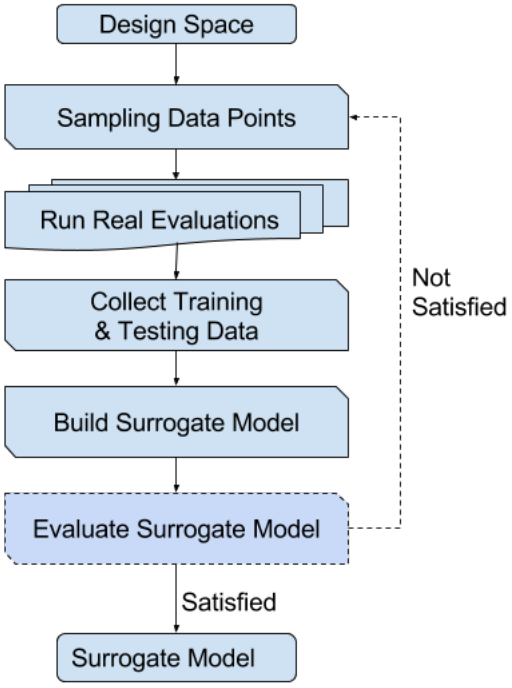

Figure 5.2: Building Surrogate Models via Offline experiments.

5.2.1 Quadratic Response Surface Model

The idea of response surface methodologies is to use polynomial approximation models which the sampled data is fitted to by a least-square regression technique. Compared to linear or higher order polynomial models, the quadratic polynomial model usually provides the favourable compromise between the modelling accuracy and the computational cost. Another advantage of Response Surface Model is that it can smooth out

the various scales of numerical noise in the data, thus it is very robust. This makes it well suited for optimisation problems in engineering design.

The construction of quadratic response surfaces requires firstly the regression surface fitting in order to obtain approximate responses, then the variances of the responses are minimised using a set of experiments (observations). The quadratic response surface model is defined as:

$$y(x) = \hat{y}(x) + \epsilon, x \in \mathbb{R} \tag{5.1}$$

where ϵ is the random error which is assumed to be normally distributed with mean zero and variance of σ^2 whereas the quadratic quadratic response surface model predictor $\hat{y}(x)$ is defined as:

$$\hat{y}(x) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \beta_{ii} {x_i}^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} x_i x_j \tag{5.2}$$

where n is the number of variables, and β_0, β_i, β_{ii} and β_{ij} are the coefficients to be determined whereas $x-i$ and y_i are the values of the independent variables. Since there are totally $p = (n+1)(n+2)/2$ unknown coefficients in Equation (5.2), at least p sample points are required to build a quadratic response surface model with n variables. The most commonly used methods to determine the coefficients are the least squares method and the gradient descent method. After the unknown coefficients are determined, the approximated response $\hat{y}$ at any untried x can be predicted by Equation (5.2).

In the literature, an application of quadratic response surface model in conjunction with an evolutionary algorithm was reported by Lian and Liou [62]. The problem was to optimise the design of the NASA rotor 67 compressor blade in order to maximise the stage pressure raise, while also minimising the entropy generation. There were 32 design variables. There were 8 design variables. The authors noted that the evaluation of 1,024 sample data points took approximately 128 hours using eight processors and a CFD simulation. Then the quadratic response surface

model was constructed using these 1,024 sampled data. They iterated an evolutionary algorithm for 200 generations with a population of 320 individuals in conjunction with the quadratic response surface model as the surrogate model to evaluate the individuals. The final solutions obtained using surrogate model were better than the reference design (baseline solution obtained by using the EA without surrogate models). The authors claimed that more computational power will reduce the time consumption.

Later on, Lian and Liou [61] used the same approach to optimise the weight of the turbine blade for the same rotor. The numerical results showed good improvement over CFD at a much reduced computational cost.

A similar approach proposed by Zhang et al. [107] was reviewed in work [35] for an aircraft wing design problem. The optimisation objective was to maximise the wing lift to load ratio and minimise the weight of the wing. They generated 100 data samples of candidate wings for training a response surface model as the surrogate model, and another 45 data samples were created to evaluate the approximation model. According to the average relative errors and the root mean squared errors, the response surface model had comparatively high accuracy. The entire optimisation took about 2 days on a personal computer of Pentium(R) 2.8GHz. The authors suggested that more concurrent computational power would reduce the time consumption.

Again in the aerodynamic field, Pagano et al. [81] applied the response surface model for a three-dimensional aerodynamic shape optimisation problem. The optimisation objectives were to maximise the aerodynamic efficiency of the propeller as well as to minimise the noise emission level. In this particular problem, the objective function calculation involved coupled computation of aerodynamics, structural behaviour and aeroacoustic, resulting in a fitness evaluation that was computationally demanding. The authors used a response surface model to explore the design space of the propeller blade, which was parameterised with 14

variables. The population size is set to 20 individuals for the evolutionary algorithm. They have obtained 20 final solutions from 340 candidate solutions based on the approximation model, all of them being better than the reference solution in terms of the two objectives.

5.2.2 Kriging Model

Different from response surface model, an alternative surrogate model is the Kriging model which can statistically predict an unknown function by minimising its mean squared error. Kriging models can be equivalent to any order of polynomials thus them work well for representing non-linear function with multiple extremes [35]. Kriging [52] method is an interpolating method which samples data from all the data points. More precisely, this method constructs probability models through the sample data and estimates the function value with a Gaussian distribution [104]. The prediction of Kriging is formed by adding up two different components as follows:

$$y(\vec{x}) = a(\vec{x}) + b(\vec{x}) \tag{5.3}$$

where $a(\vec{x})$ stands for the expected value of the real function. This function can be modelled in different ways. In the case of polynomials, it is defined as:

$$a(\vec{x}) = a_0 + \sum_{i=1}^{L} \sum_{j=1}^{R} a_{ij}(x_i)^j \tag{5.4}$$

where R is the polynomial order with L dimensions (design variables). The term $b(\vec{x})$ in Equation (5.3) is a Gaussian random function with zero mean and non-zero covariance. This term stands for a localised deviation from the global model, in other words, it represents the influence of every data point over the global model. The general form of $b(\vec{x})$ is a weighted sum of N functions and it can be expressed as:

$$b(\vec{x}) = \sum_{n=1}^{N} b_n K(h(x, x_n)) \tag{5.5}$$

where b_n are the weights to be determined and $K_n(x)$ is a set of covariance functions between the n^{th} data point and a random point x. It is defined as:

$$h(x, x_n) = \sqrt{\sum_{i=1}^{L} \left(\frac{x_i - x_{in}}{x_i^{max} - x_i^{min}} \right)^2} \tag{5.6}$$

where x_{in} is the i^{th} data point, x_i^{max} and x_i^{min} respectively represent the upper and lower bounds of the search space.

In the literature, the study of Kriging model can be found in various problems industrial problems. D'Angelo and Minisci [16] used an evolutionary algorithm in conjunction with a Kriging model to solve a multi-objective optimisation problem. The objectives of the optimisation problem were to minimise the drag force coefficient and lift force coefficient simultaneously. The authors used a Kriging model to search the design space of subsonic airfoils. The design space is parameterised with 5 variables and is subject to the extreme values of the objectives. They iterated the a population with 100 individuals for 150 generations. During the iteration, the Kriging model was used along with an evolutionary control technique which was adopted to avoid the evolution algorithm to converge to a false optima. In their case, they evaluated a subset of the individuals using the real fitness function to enrich the correct solutions database, which was the basis of the learning procedure for the surrogate model. The authors claimed that the use of the approximation model as well as the evolution control technique together significantly increased the speed of converging. The final results showed that only 2,300 real evaluations were needed using the surrogate whereas 2910 real evaluations were needed without the surrogate.

Song and Keane [94] presented a study of using a Kriging model to reduce the computational cost of a multi-objective optimisation problem. The main goal of their study is to identify the tradeoff between aerodynamic performance and noise effects for the nacelle of a civic aircraft. The geometry was parameterised using 40 parameters, and 33 of them were considered as design variables. Since they applied the CFD us-

ing a commercial software to perform the real fitness evaluation, the computational cost was tremendous. Thus they constructed a Kriging model to approximate the fitness values in order to keep the usage of real CFD evaluations at a minimum level. Similar to work [16], Song and Keane [94] also adopted an evolution control technique in a way that the Kriging model was continuously updated using new samples with good fitness obtained by the Kriging model and the CFD evaluations. The authors argued that even 33 design variables could cause the evolutionary algorithm to have difficulties to converge. Then they presented a set of promising results by reconstruct the Kriging model only using 7 variables. They reported that the reduced Kriging model contributed to the finding of a similar solution in terms of fitness at a significantly less computational cost. This simplification indicates that Kriging model cannot handle a large number of design variables.

5.2.3 Artificial Neural Networks

The widespread application of Artificial Neural Networks (ANN) in many research fields is primarily depend on their capability to approximate complex nonlinear mappings directly from the input samples [40]. An ANN consists of a set of processing elements, also known as neurons or nodes, which are interconnected. Training an ANN is typically accomplished using examples to adjust the connection weights between nodes. By the way ANN is trained, they can roughly be divided into supervised, unsupervised, and reinforcement learning. Supervised learning mean that the connecting weights are adjusted based on direct comparison between the output of an ANN and the actual output of the training example [34]. A gradient descent-based optimisation algorithm such as backpropagation [38] is widely used to adjust connection weights in the ANN iteratively in order to minimise the error.

Among many kinds of neural networks, the feedforward neural networks have been investigated most thoroughly. In many applications, feedforward neural networks have been used as approximate models, al-

though it is well known that the approximation accuracy strongly depend on their architecture (i.e. number of hidden layers and nodes of each hidden layer). The complexity of the structure also determines the learning efficiency of neural networks [42].

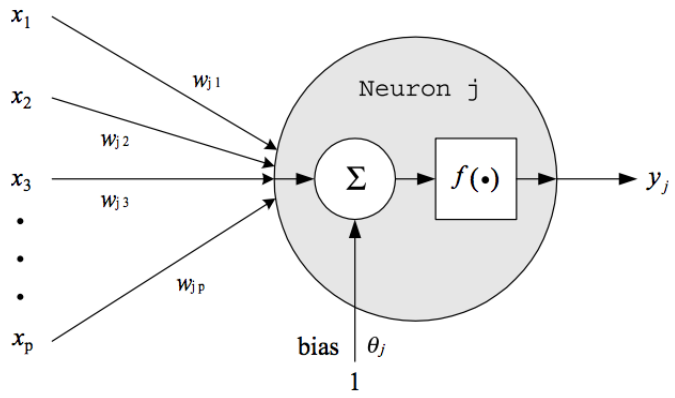

Figure 5.3: One node of MLP: an artificial neuron.

A node in a MLP refers to an artificial neuron. As shown in Figure 5.3, the node j computes the weighted sum of the inputs at the presence of a bias, then passes this sum through an activation function. This computation can be expressed as:

$$\begin{aligned} v_j &= \sum_{i=1}^{P} w_{ij} x_i + \theta_0 \\ y_j &= f_j(v_j) \end{aligned} \tag{5.7}$$

where v_j is the linear combination of inputs $x_1, x_2, \ldots, x_p$, w_{ij} are the connection weights between the input x_i and the neuron j, θ_j is the bias, $f_j(\cdot)$ is activation function of the neuron j and y_j is the output.

The schematic representation of a simple feedforward neural network with one input layer, one hidden layer and one output layer is shown in Figure 5.4. The hidden layer consists of many neurons. A MLP can be defined as:

$$Y(x) = \sum_{j=1}^{K} w_j f_j \left(\sum_{i=1}^{P} w_{ij} + \theta_j \right) + \theta_0 \tag{5.8}$$

where P is the number of inputs $x_1, x_2, \ldots, x_p$ (design variables), K is the

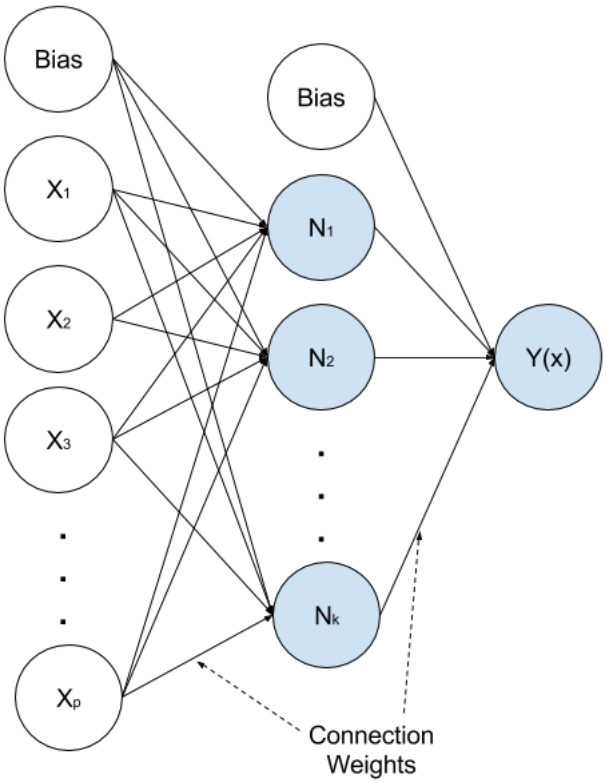

Figure 5.4: Architecture of a multilayer perceptron network.

number of hidden nodes, w_j and w_{ij} are the connection weights to be learned and θ_j and θ_0 are the bias to be determined. Please note that the architecture of MLP shown in Figure 5.4 is for numeric prediction. The commonly used activation function is the sigmoid function and it is defined as:

$$f(a) = \frac{1}{1 + e^{-a}} \tag{5.9}$$

where c is a constant. If the bias term θ_j is positive then the sigmoid function shifts to the left and vice versa.

The idea of the network is simple. When data (i.e. a the Cartesian coordinates of wind turbines inside a wind farm) are presented at the input layer, the network nodes perform computation in the successive layers until an output value (i.e. energy output) is obtained at the output

node.

Arabnia and Ghaly [2] described an effective and practical optimisation strategy make use of genetic algorithm in conjunction with an artificial neural network to improve the turbine efficiency. The authors used the E/TU-3 turbine to setup the blade geometry and used it as the reference design to compare the optimisation results to. The optimisation objectives are the maximisation of the isentropic efficiency for the stage and the minimisation of the stream-wise vorticity. There were 5 design variables for the blades. The ANN model had a single hidden layer structure with 50 hidden nodes, and it was trained based on 23 CFD simulations. The paper indicates that a single CFD evaluation took approximately 10 hours. The optimisation process used the trained ANN model to estimate the fitness values of a population with 50 individuals for 150 generation. At the end, the best obtained results were re-evaluated using the CFD evaluation. The paper claims that the obtained designs attained 1.2% improvement of stage efficiency in comparison to the reference turbine design. Considering the limited amount of training data, that is a considerable refinement without high computational cost.

Alonso et al. [1] presented a set of procedures to optimise the design of a supersonic aircraft. The optimisation focused the minimisation of the sonic boom loudness and the maximisation of the aircraft range. The authors trained an artificial neural network as the surrogate model which was a single hidden layer perceptron with sigmoid activation functions. The ANN was fitted with 300 data samples, then 150 more data samples were used to test the ANN. Different from work [2], the training and testing samples were obtained using low accuracy simulations rather than the CFD evaluation to reduce computational cost. The optimisation is accomplished by varying the values of 10 design variables. The searching process used the trained ANN model to estimate the fitness values of a population with 64 individuals for 1,000 generations. In order to save computational cost, the real fitness evaluation was used to do local search after the surrogate model found a promising region in the design space. The results show that these procedures are computationally less

expensive, thus are suitable to constitute a set of valid initial designs during the preliminary design phase, then the initial designs can be used for continued refinement.

Rai [86] reported an approach to optimise robust design of a turbine blade airfoil shape taking into account the performance degradation due to the uncertainties during manufacturing. The aim of the optimisation was to minimise the variance of the pressure distribution over the surface of the airfoil as well as to maximise the wedge angle at the trailing edge. In the fitness evaluation, the manufacture tolerance is obtained by introduce random noise to the blade geometry whereas the surface pressure distribution is calculated using a CFD simulation. The blade geometry is defined by 8 parameters, but only 2 of them are varied during the optimisation. In order to reduce the computation cost caused by using CFD simulations, the authors constructed a hybrid neural network as the approximation model which comprise of 10 individual single hidden layer feedforward networks. The evolutionary optimisation process searches for 25 generations with a relatively small population size of 10 candidate solutions. The authors claim that their hybrid neural network construction approach is able to obtain accurate surrogate model thus the overall optimisation process can done at a lower computational cost.

5.2.4 Discussion of Different Surrogate Models

Among the surrogate models, the regression models such as the quadratic response surface model works well for the optimisation problems with relatively less design variables. The Kriging models are able to represent nonlinear, multimodal functions, thus they are well suited for relatively complicated design space. However, the use of an artificial neural network can yield more accurate approximation models for highly nonlinear design space under the same set of constraints when compared to the ordinary kriging method [15].

As previously mentioned, CFD simulation is commonly used in many

real world applications, as the CFD is one of the most accurate ways to evaluate aerodynamic related problems such as aircraft wing design, turbine blade airfoil design and wind farm layout design. This is the reason that the literature reviews are primarily focused on building surrogate models to approximate the CFD simulations. But the WFLOP takes into account significantly more design variables compared to the reviewed studies. For example, in a two-dimensional planner version of the WFLOP (in which all turbines are assumed identical), one hundred turbines require at least 200 design variables under the minimum amount of constraints. As we reviewed, both regression methods and feedforward artificial neural networks showed promising performance in many highly computational complex results. Due to the complexity of the WFLOP, the feedforward artificial neural networks and the regression models are more likely to accurately approximate the real fitness function at a lower computational cost.

In the literature, Jin et al. [48] suggested a simple structure of single hidden layer MLP with small number of hidden units. In work [17], tree-based regression models (i.e. M5P) performed at least as well as or better than traditional black-box models (i.e. MLP). In this work, we conducted a series of initial offline experiments in Chapter 6 to determine which surrogate model can accurately approximate the real fitness function of the WFLOP.

5.3 Scalability Issues of Surrogate-assisted EAs

In the literature, there are many surrogate-assisted evolutionary algorithms have been developed. These approaches have been showing promising in reducing the computational cost in many computational expensive optimisation problems. However, most of them work only for relatively low-dimensional (i.e. small number of design variables) optimisation problems [96]. As can be seen in Table 5.1, the dimension of optimisation problems reviewed in Section 5.2 is very limited. According to Sun et al. [96], in the literature the maximum dimension solved by surrogate-

assisted EA is 50, which was reported in work [64]. The author proposed a Gaussian process based surrogate model assisted evolutionary algorithm in conjunction with dimension reduction techniques to solve medium-scale computational expensive problems.

Publication	Design Variables
Lian and Liou [62]	32
Zhang et al. [107]	8
Pagano et al. [81]	14
D'Angelo and Minisci [16]	5
Song and Keane [94]	7
Arabnia and Ghaly [2]	5
Alonso et al. [1]	10
Rai [86]	2

Table 5.1: The dimension of optimisation problems reviewed in Section 5.2.

Due to high input dimension of the objective function (i.e. large number of design variables), the constructed surrogate models may not be able to accurately approximate the real fitness function. During the optimisation process, poor surrogate models can not provide accurate fitness prediction. An inaccurate surrogate model may introduce false optimums, which mislead the evolutionary search [45]. Thus the idea of constructing global surrogate models for relatively high-dimensional problems (i.e. WFLOP) is very challenging.

5.4 The Management of Fitness Approximation in EA

5.4.1 A Brief Review on Managing Surrogates

An issue pointed out by Jin [45] is that if the optimisation is solely based on a surrogate, then there is a risk that an inaccurate surrogate cannot provide sufficiently accurate fitness values. This issue will introduce false optima that do not exist in the original problem, thus results in mislead-

ing the evolutionary algorithm. The surrogate management in surrogate-assisted evolutionary optimisation addresses this issue, in a way that the surrogates must be used together with the real fitness function [47].

The use of surrogate-assisted local search approach is common in both single-objective and multi-objective evolutionary optimisation problems. The results reported in works [79, 108, 71] have shown that using multi-surrogate models can accelerate the convergence to good solutions on a limited computational budget.

Surrogates can be widely applied to different parts of an EA. For example, poor solutions generated during population initialisation, mutation and crossover can be filtered out by using surrogates [46]. More recently, an aggregated surrogate was built and used to pre-screen candidate solutions based on estimated fitness values, before the real fitness evaluation [65].

Techniques for managing surrogates for fitness evaluation in evolutionary algorithms can be divided into individual-based, generation-based and population-based [45]. By individual-based, the real-fitness function is used for fitness evaluation for some individuals in a generation [48, 10]. Contrary to that, generation-based approaches use the surrogate for fitness evaluation in some the the generations whereas the real fitness function is used in the rest of the generations [87, 63]. Jin [45] claims that in population-based surrogate model management techniques, more than one sub-population co-evolves, each using its own surrogate for fitness evaluation.

5.4.2 Pre-Selection Surrogate Management Strategy

Two closely related techniques for managing surrogates for fitness evaluations are reported by Jin [46]. The idea behind these strategies is to re-evaluate the individuals that have a promising fitness, and the higher accuracy of the surrogate estimation, the better the chance that a good solution will be found. The first strategy in question is called Pre-selection

Strategy, which is similar to individual-based methods. As shown in Figure 5.5, in a (μ, λ) evolutionary algorithm using pre-selection strategy, $\lambda^\star \geq \lambda$ offspring are generated and then evaluated using the surrogate. After that, the λ best individuals are chosen to be re-evaluated using the real fitness function. As a result, all the μ parents for breeding are chosen based on real fitness values.

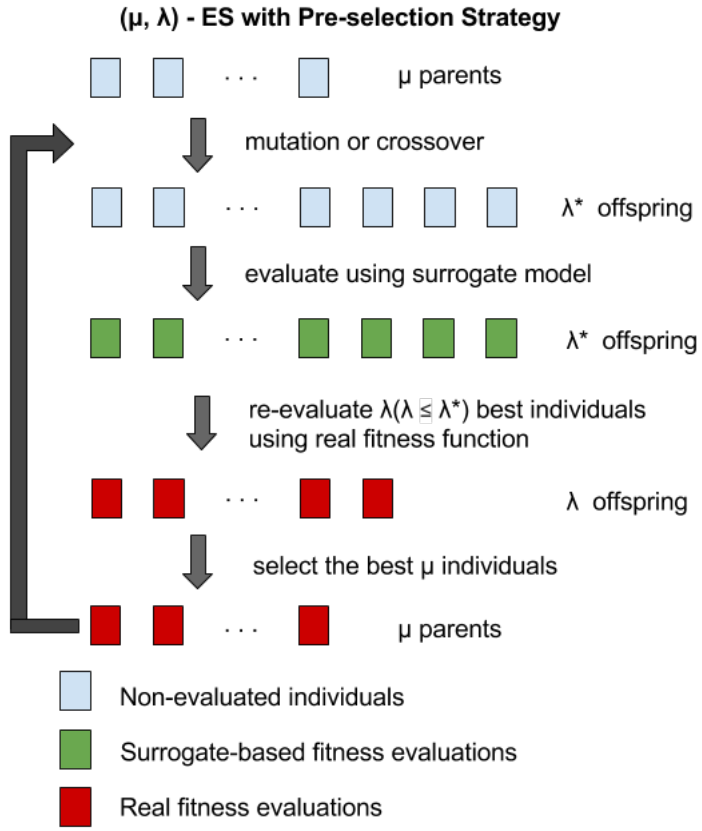

Figure 5.5: Pre-selection Surrogate Management Strategy [46].

5.4.3 Best Surrogate Management Strategy

Another strategy is the Best Strategy which is shown in Figure 5.6. Contrary to the Pre-selection Strategy, at the beginning all λ offsprings are evaluated using the surrogate. Then $\lambda^\star \leq \lambda$ best individuals are chosen based on estimated fitness values for re-evaluation using the real fitness

function. Inevitably, there is a chance that some of the μ parents are chosen based on surrogate produced fitness values.

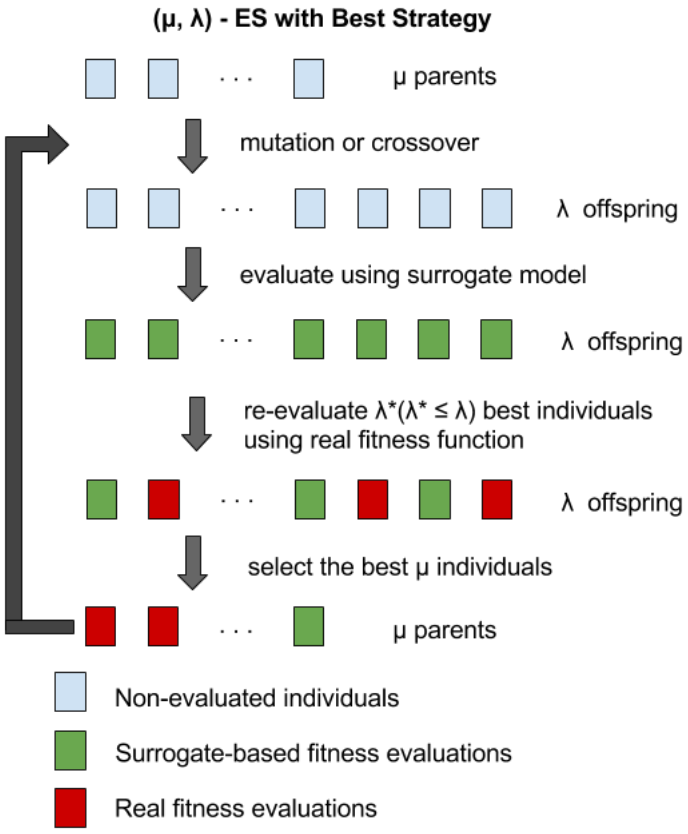

Figure 5.6: Best Surrogate Management Strategy [46].

5.4.4 Modified Pre-Selection and Best Surrogate Management Strategy

Based on successful real world applications [94, 68] from the literature, the evolution control technique continuously updates the surrogate models shows promising results. The basic idea is to add new samples with good fitness obtained by the surrogate model and the real fitness evaluations to the training data. Thus the surrogate model can be re-built based on more training data during the optimisation process. In general, a larger amount of training data means better approximation accuracy [8]. With this concept in mind, we adopted a surrogate model re-training

evolution control technique in the Pre-Selection and Best surrogate management strategies.

Figure 5.8 and Figure 5.9 show how surrogates are managed after the adopting of a surrogate model re-training technique. Specifically, at the beginning of the algorithm is the complete random initialisation of μ parents to generate λ offspring. The evolutionary process is managed based on the total number of real fitness function calls. In this project the maximum real fitness evaluations is 2,000. During the first 1,000 real calls, only the real fitness function is used to evaluate the individuals. Meantime, training data is collected in the format of raw Cartesian coordinates based on the real fitness value during this process. As shown in Figure 5.7, the attribute values of the training data are converted from Cartesian coordinates (x_i, Y_i) to polar coordinates (d_i, θ_i), which are then sorted according to the distances d between the turbines and the zero point. These layouts (polar coordinates) are added into the training data along with there real fitness value. Then a surrogate model is trained using the collected training data.

Then, the evolution procedure continues in conjunction with the surrogate model. As shown in Figure 5.8, in the Pre-selection strategy, a number of $\lambda^\star \geq \lambda$ offspring are generated in each generation. These $\lambda^\star$ offspring are first evaluated using the surrogate model, then, based on surrogate fitness values, the number of λ best individuals are chosen to be re-evaluated using the real fitness function. At the same time, new training data is collected from the re-evaluated individuals. The newly added raw layouts are transferred into sorted polar coordinates as well (see Figure 5.7). If runout of the rest 1,000 real fitness function calls, because of the nature of Pre-selection strategy, the best individual is always selected based on the real fitness values.

Contrary to that, in the Best strategy, only $\lambda^\star \leq \lambda$ best individuals are chosen to be re-evaluated using the real fitness function based on surrogate fitness values. At the same time, new training data is collected from the re-evaluated individuals. If the 1,000 real fitness function evaluations

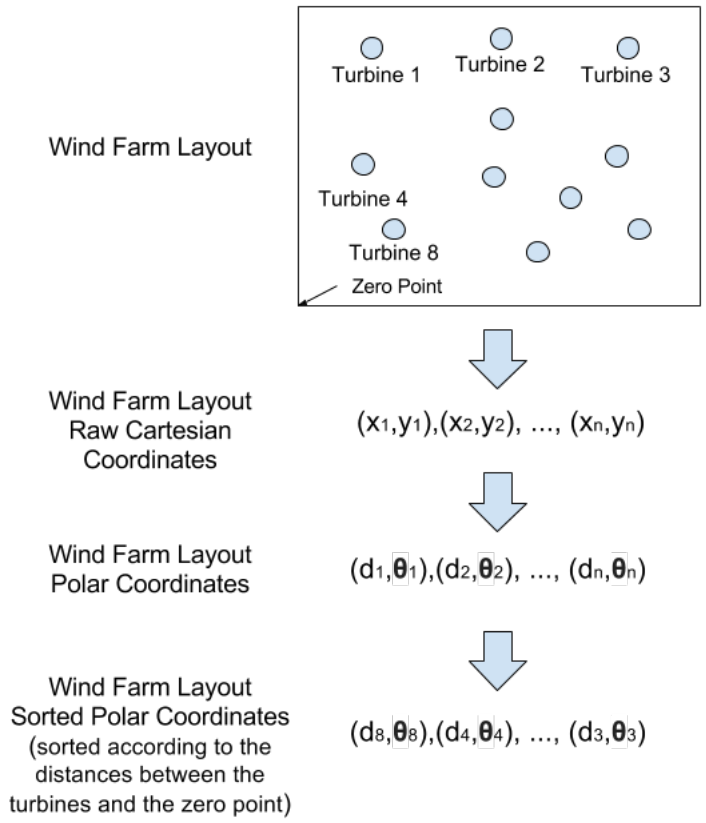

Figure 5.7: Collect and prepare wind farm layout data. The layout data consists of raw Cartesian coordinates (x_i, Y_i) of the wind turbines. Then they are converted into polar coordinates (d_i, θ_i). At the end the polar coordinates are sorted according to the distances d between the turbines and the zero point.

limit is exceeded, the best individual is selected based on fitness value regardless of real or estimated, then it's re-evaluated using real fitness function. It is important to note the difference at the final selection between the Pre-selection and Best strategy. Because there is a chance that a good individual might be chosen based on its estimated value (denoted as green box in Figure 5.9), which may result in disappointment findings (i.e. false optimum). Thus it has to be re-evaluated using the real fitness function at the very end.

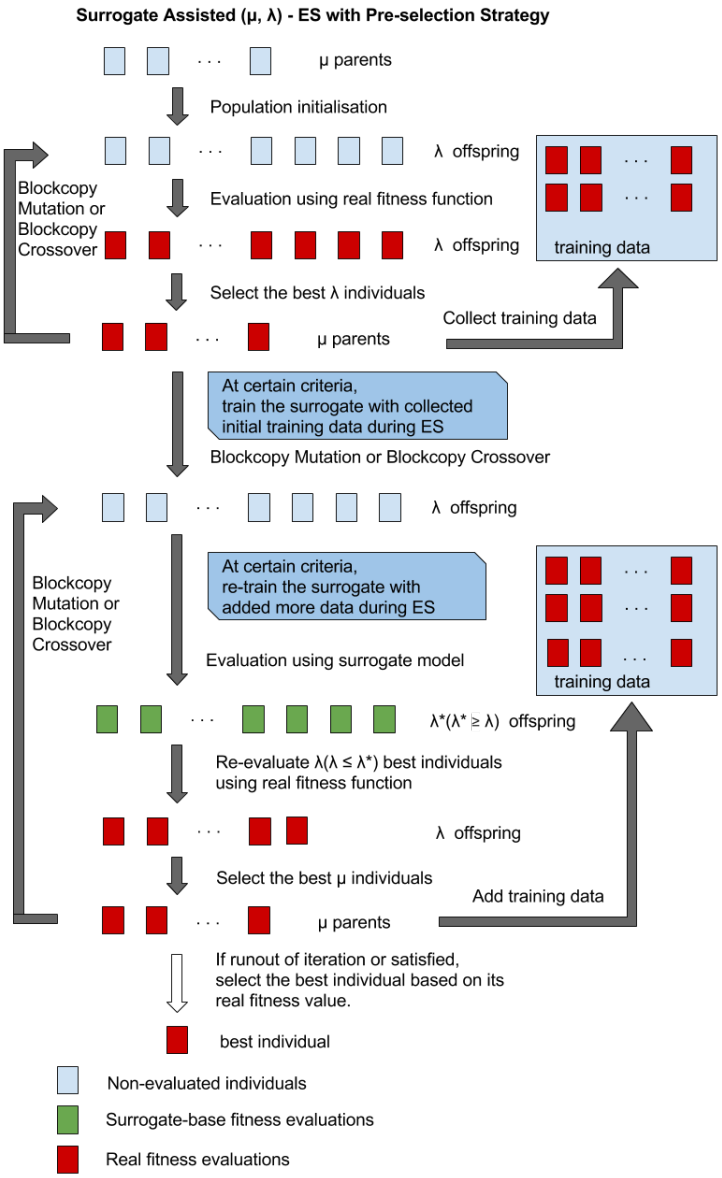

Figure 5.8: Surrogate Assisted (μ, λ) - ES Using Pre-selection Strategy Strategy with Surrogate-Retraining.

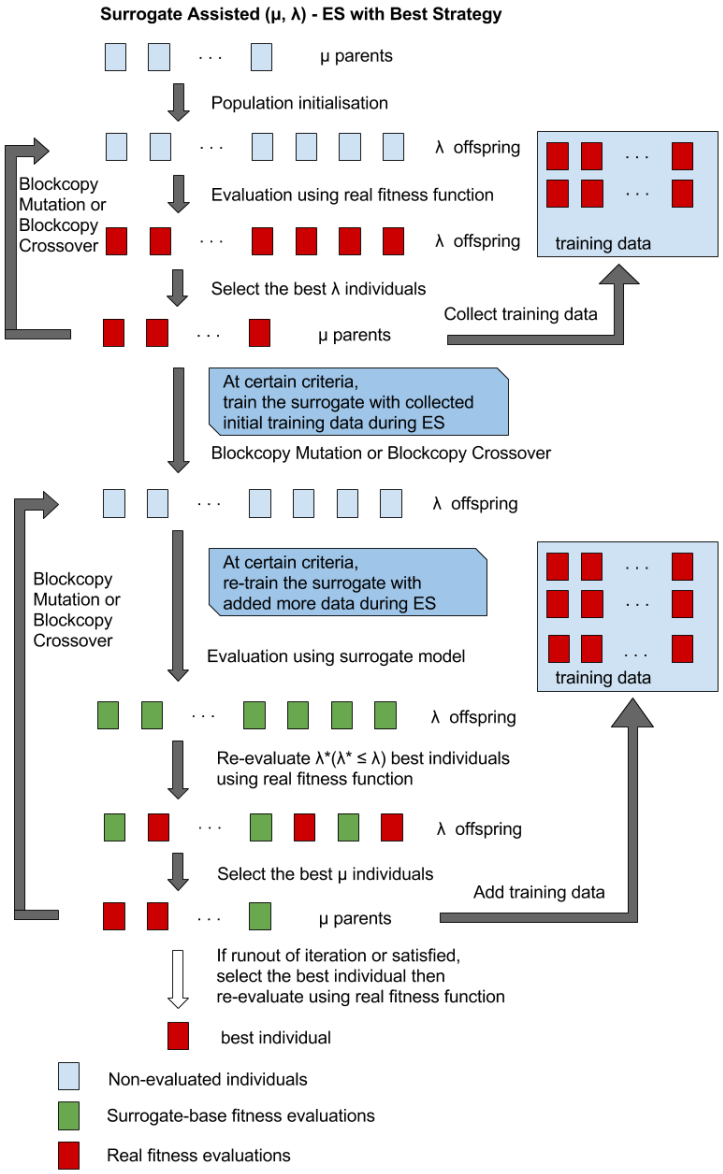

Figure 5.9: Surrogate Assisted (μ, λ) - ES Using Best Strategy Strategy with Surrogate-Retraining.

Chapter 6

Initial Experiments: Comparison of Different Approximation Models

6.1 Approximation Accuracy Measurements

This chapter illustrates a set of experiments using different data mining algorithms to approximate the real fitness function. The design of our experiments is based on Figure 5.2. As previously mentioned, the accuracy of the approximation is one of the main issues in the design and use of surrogate assisted approaches. In this case, the surrogate should be able to ensure the selection of the best individuals in terms of the original fitness function.

It is difficult to compare the different types of approximation as performance is problem dependent. Jin [45] suggested that approximation models should start from simple ones, for example, lower order polynomial models, to see if the given training and testing data can be fitted with reasonable accuracy. If the simple model fails to fit high dimensional problems, then higher order polynomial or neural networks should be considered.

There are many ways for measuring the error between real fitness values and surrogate estimated fitness values [34]. In this work, since the surrogate is used to predict numeric values, we are particularly inter-

ested in the correlation coefficient and root relative squared error metrics.

We denote the true value of interest as θ and the value estimated using some algorithm as $\hat{\theta}$. The correlation coefficient of N pairs of true and estimated values is written as:

$$r_{\theta\hat{\theta}} = \frac{\sum_{i=1}^{N}(\theta_i - \overline{\theta})(\hat{\theta} - \overline{\hat{\theta}})}{\sqrt{\sum_{i=1}^{N}(\theta_i - \overline{\theta}) \sum_{i=1}^{N}(\hat{\theta}_i - \overline{\hat{\theta}})}} \tag{6.1}$$

where $\overline{\theta}$ denotes the average of real fitness values and $\overline{\hat{\theta}}$ denotes the average of estimated fitness values.

The root relative squared error is written as:

$$E^{rrse} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{\theta}_i - \theta_i)^2}{\sum_{i=1}^{N}(\overline{\theta} - \theta_i)^2}} \tag{6.2}$$

where $\overline{\theta}$ denotes the average of real fitness values.

Here is the interpretation of these measurements. The correlation coefficient shows how much θ and $\hat{\theta}$ are related. As shown in Equation (6.1), the coefficient value is given between -1 and 1, where 1 indicates a very strong linear relation (i.e. the value of θ gets large, $\hat{\theta}$ gets large). The 0 correlation coefficient value means there is no relation between original fitness value and predicted fitness value. The -1 value tells that θ and $\hat{\theta}$ are linear related inversely (i.e. the value of θ gets large, $\hat{\theta}$ gets small). In other words, correlation is positive when the values increase together, and correlation is negative when one value decreases as the other increases.

Equation (6.2) statistically compares true values to their estimates. In Equation (6.2), the values of $\sum_{n=1}^{N}(\overline{\theta} - \theta_i)^2$ represent how much θ differs

from its mean value $\overline{\theta}$. The values of $\sum_{n=1}^{N}(\hat{\theta}_i - \theta_i)$ represent the squared differences, which are then divided by the variation of θ so that they have a scale from 0 to 1, and this value is multiplied by 100 we can get similarity in 0-100 percentage. An advantage of using square roots in the metric is that the extreme values have more influence on the result [34].

6.2 Initial Offline Experiment Data Description

The Weka Experiment Environment [34] enables the user to create, run, modify, and analyse experiments in a more convenient manner that is possible when processing the data mining algorithms individually. For example, the user can create an experiment that runs several schemes against a series of datasets and then analyse the results to determine if one of the schemes is (statistically) better than the other schemes.

In this thesis, we are interesting in finding which approximation model against a series of datasets can achieve the minimum root relative squared error, meanwhile, the model that has the highest correlation coefficient. The experiment data is collected from the runs of my implementation for WFLOP on wind farm scenarios from literature (more detail of the scenarios are given in Chapter 7).

The wind farm layout datasets used in these initial experiments are generated based on four benchmark wind farm scenarios that are selected from the recent literature (see Section 7.1). The datasets are generated using a biased random walk (i.e. evolutionary strategy). During the optimisation process, layouts are collected along with there real fitness values. In total, there are 2,000 layouts (raw Cartesian coordinates) evaluated by the real fitness function. As shown in Figure 5.7, the attribute values of the training data are converted from Cartesian coordinates (x_i, Y_i) to polar coordinates (d_i, θ_i), which are then sorted according to the distances d between the turbines and the zero point. These layouts (polar coordinates) are added into the training data along with there real fitness value. We generated 10 sets of layouts data for each wind farm

scenario, each set consists of 2,000 layouts along with their fitness values.

In our project, the data file is in the Attribute-Relation File Format (ARFF) [34]. The ARFF attributes are either the two-dimensional Cartesian coordinates or the polar coordinates of turbines inside the wind farm and the cost of energy of the wind farm, both of them can are defined as:

$$
\begin{aligned}
&@attribute < coordinate\ d_i >< numeric > \\
&@attribute < coordinate\ \theta_i >< numeric > \\
&\qquad\qquad \vdots \\
&@attribute < cost\ of\ energy >< numeric >
\end{aligned}
$$

where $i \in N$ is the number of turbine.

In the ARFF data section, each instance is represented on a single line, and the attribute values for each instance are delimited by commas. They must appear in the order that they were declared in the header section. Here is an example of an instance entry using polar coordinates:

$$
\begin{aligned}
&@data \\
&2871.722283354609, 0.92578665166, \ldots, 0.0014044201
\end{aligned}
$$

where the numeric values "2871.7222..." and "0.9257..." respectively represent the distance and angle of the turbine inside the wind farm from the zero reference point, and the numeric value "0.0014..." is the value of cost of energy (fitness value). Please note that the 12 digits after the decimal point is an extension of the default 6 digits. This modification is better for optimum accuracy.

The random projector filter from Weka was also used to generate features for this experiment [34]. It is used to reduces the dimensionality of the data by projecting it onto a lower dimensional subspace using a random matrix with columns of unit length (i.e. It will reduce the number of attributes in the data while preserving much of its variation, but at a

much less computational cost). The configurations of number of turbines and number of attributes for each scenario after each filter is applied are shown in Table 6.1.

Scenario vs. Filter	Kusiak & Song 1	Kusiak & Song 2	2014 Comp 1	2014 Comp 3
Random Projection	100 turbines 50 attributes	100 turbines 50 attributes	220 turbines 100 attributes	710 turbines 200 attributes
Polar	100 turbines 200 attributes	100 turbines 200 attributes	220 turbines 440 attributes	710 turbines 1420 attributes
Raw	100 turbines 200 attributes	100 turbines 200 attributes	220 turbines 440 attributes	710 turbines 1420 attributes

Table 6.1: The number of turbines and number of attributes for each scenario after each filter is applied.

6.3 Initial Offline Experiment Setup

The task of the surrogate in this work is to predict numeric values. A popular choice for that is the M5 algorithm which was originally invented by Quinlan [85] and Wang and Witten [101] made improvements. The M5 algorithm learns efficiently and can tackle tasks with very high dimensionality (i.e. up to hundreds of attributes). Weka implemented a tree-based version of the M5 algorithm as the classifier M5P [34], which combines a conventional decision tree with linear regression functions at the nodes. A very recent study [17] reports that comprehensible models (i.e. M5P tree-based regression model) perform at least as well as or better than traditional black-box models (i.e. MLP neural network). So we employed a MLP as well as M5P as the surrogate models for the WFLOP fitness function approximation.

In [45] Jin suggested that approximation models should start from simple ones, for example, lower order polynomial models, to see if the given training and testing data can be fitted with reasonable accuracy. If the simple model fails to fit high dimensional problems, then higher order polynomial or neural networks should be considered.

Our initial experiment was to compare the performance in terms of correlation coefficient and root relative squared error between M5P and MLP against a series of wind farm layout datasets with different feature types in the Weka Experiment Environment [34]. The detail of the major options for each classifier used in the experiment (some options are set to default value) are shown in Table 6.2 and Table 6.3. Please note that the 4 different values of option -H or -M are the settings that are utilised. The results can provide an idea of what features to use and which classifier configuration to use later on for the surrogate models in the surrogate-assisted evolutionary algorithm.

The wind farm layout datasets are prepared through three filters including raw cartesian coordinates, sorted polar coordinates (transformed from raw cartesian coordinates, see Figure 5.7), random projections. The experiment type is Regression and the Train/Test Percentage Split option is set to 50% train. Both classifiers are given one run by using the four sets of different options (-H or -M values) on four scenarios with ten different datasets through three data filters. In total, there are 2 classifiers $\times$ 4 sets of different classifier options (see Tables 6.2 and 6.3) $\times$ 4 scenarios (see Table 7.1) $\times$ 10 different datasets (2,000 layouts each) $\times$ 3 data filters (see Table 6.1) = 960 experimental runs.

Classifier	Option	Value	Description
MLP	-H	5,10,20,50	The hidden layers to be created for the network.
	-L	0.3	Learning Rate for the backpropagation algorithm.
	-M	0.2	Momentum Rate for the backpropagation algorithm.
	-N	500	Number of epochs to train for.
	-V	0	Percentage size of validation set to use to terminate,training.
	-E	20	The consecutive number of errors allowed for validation, testing before the network terminates.

Table 6.2: Options for the MLP classifier used in our initial offline experiments.

Classifier	Option	Value	Description
M5P	-M	5,10,20,50	Minimum number of instances per leaf.
	-U	False	Use unsmoothed predictions.
	-N	False	Use unpruned tree/rules.
	-L	-1	Maximum tree depth (default -1, no maximum).

Table 6.3: Options for the M5P classifier used in our initial offline experiments.

6.4 Initial Offline Experiment Results

Tables 6.4 to 6.7 illustrate the statistical analysis results obtained from the Weka Experiment Environment. The table row header indicates the features used during learning whereas the column header shows the classifier with the key option. In the cells, the annotation v or $*$ indicates that a specific result is statistically better (v) or worse (*) than the baseline scheme. In this case, the baseline is set by the first classifier configuration (MLP -H 5) of each table.

As can be observed, the performance are quite different for each classifier using various options with different filters. Overall, both MLP and M5P failed on the dataset applied random projection filter. This is mainly because reducing the dimensionality of the wind farm layout data results in low accuracy. On the other hand, both M5P and MLP delivered good results using raw Cartessian and polar coordinates.

It is clear that M5P (-M 5) classifier using polar coordinates outperformed the rest of classifiers across four scenarios in terms of higher correlation coefficient and lower root relative squared error. On the simpler scenarios Kusiak & Song 1 and 2, the M5P classifiers have showed higher learning capability than the MLP classifiers. But the MLP classifiers showed similar learning efficiency on difficult scenarios such as 2014 Comp 1 and 3. Particularly in scenario Comp 3, this is the biggest scenario among the four given scenarios where M5P (-M 5) and MLP (-H 20) showed similar performance using sorted polar coordinates. According to the results, the M5P (-M 5) is a better surrogate choice for the surrogate-assisted evolutionary algorithm. But we also employed the MLP (-H 20) because of its promising performance in the literature.

Classifier vs. Feature	MLP -H 5	MLP -H 10	MLP -H 20	MLP -H 50	M5P -M 5.0	M5P -M 10.0	M5P -M 20.0	M5P -M 50.0
Random Projection	0.18 143.97	0.21 170.66	0.23 180.27	0.22 176.08	0.19 99.78	0.16 102.18	0.16 102.18	0.14 101.65
Polar (sorted)	0.40 129.74	0.53 v 101.80 *	0.58 v 88.24 *	0.60 v 87.72 *	0.84 v 54.58 *	0.83 v 56.05 *	0.81 v 59.11 *	0.74 v 67.38 *
Raw	0.36 134.46	0.44 112.27	0.53 v 95.51 *	0.53 v 93.60 *	0.85 v 53.56 *	0.83 v 56.05 *	0.82 v 58.26 *	0.76 v 65.43 *

Table 6.4: Comparison of correlation coefficient (first line in each cell) and root relative squared error (second line in each cell) on datasets for wind farm scenario Kusiak & Song 1 [54].

Classifier vs. Feature	MLP -H 5	MLP -H 10	MLP -H 20	MLP -H 50	M5P -M 5.0	M5P -M 10.0	M5P -M 20.0	M5P -M 50.0
Random Projection	0.19 146.73	0.20 178.31	0.22 183.75 v	0.20 173.46	0.18 99.57 *	0.17 100.95 *	0.15 101.92 *	0.13 101.43 *
Polar (sorted)	0.36 137.56	0.46 v 109.88 *	0.52 v 95.59 *	0.54 v 94.26 *	0.80 v 59.99 *	0.79 v 61.69 *	0.77 v 63.21 *	0.72 v 69.49 *
Raw	0.36 135.55	0.44 110.89 *	0.51 v 97.63 *	0.53 v 95.38 *	0.79 v 61.09 *	0.78 v 62.92 *	0.76 v 64.71 *	0.71 v 70.50 *

Table 6.5: Comparison of correlation coefficient (first line in each cell) and root relative squared error (second line in each cell) on datasets for wind farm scenario Kusiak & Song 2 [54].

Classifier vs. Feature	MLP -H 5	MLP -H 10	MLP -H 20	MLP -H 50	M5P -M 5.0	M5P -M 10.0	M5P -M 20.0	M5P -M 50.0
Random Projection	0.37 122.03	0.36 146.32	0.37 154.58	0.33 152.82	0.37 93.90	0.34 95.81	0.30 97.38	0.28 97.60
Polar (sorted)	0.81 62.52	0.87 v 50.46 *	0.89 v 46.84 *	0.86 52.39	0.90 v 42.50 *	0.89 v 44.82 *	0.89 v 46.16 *	0.86 50.58
Raw	0.82 61.32	0.86 52.46	0.87 50.89 *	0.85 55.78	0.90 43.12	0.89 45.26	0.89 46.38	0.87 48.22

Table 6.6: Comparison of (first line in each cell) and root relative squared error (second line in each cell) on datasets for wind farm scenario 2014 Comp 1 [105].

Classifier vs. Feature	MLP -H 5	MLP -H 10	MLP -H 20	MLP -H 50	M5P -M 5.0	M5P -M 10.0	M5P -M 20.0	M5P -M 50.0
Random Projection	0.45 127.41	0.60 v 92.23 *	0.66 v 81.99 *	0.60 94.75	0.36 96.29 *	0.35 97.06 *	0.31 98.21 *	0.27 * 99.40 *
Polar (sorted)	0.92 39.30	0.93 35.31	0.94 34.49	0.89 47.52	0.95 30.63	0.94 32.47	0.94 34.39	0.93 37.24
Raw	0.87 49.62	0.88 47.37	0.86 51.48	0.89 45.19	0.92 38.85	0.91 40.98	0.90 43.34	0.87 48.09

Table 6.7: Comparison of (first line in each cell) and root relative squared error (second line in each cell) on datasets for wind farm scenario 2014 Comp 3 [105].

Chapter 7

Evaluation and Results

This chapter describes the wind farm scenarios and the real fitness function that are used in our evaluation in more detail. Then we present the experimental design used to compare different surrogate models for the WFLOP. We implemented the surrogate-assisted (μ, λ) evolutionary algorithms using BlockCopy operators with the Best and Pre-selection surrogate management strategies. We also used the M5P and MLP algorithm implementations available from the Weka [34] platform. The experimental aim is to discover whether surrogate-assisted evolutionary strategies are more effective on the WFLOP than traditional evolutionary algorithms.

7.1 Wind Farm Scenarios

We selected four benchmark wind farm scenarios from the recent literature. The first two simpler scenarios are proposed by Kusiak & Song in [54]. Kusiak & Song's first scenario is an artificial example in which wind blows predominantly in a single direction while the second scenario is a more realistic one that describes the wind speed direction distribution at an actual industrial wind farm [54]. In both scenarios, the size is limited to 4km by 4km and the number of turbines is fixed at 100.

Scenario Comp1 and Comp3 are selected from the 2014 Wind Farm

Layout Optimisation Competition [105]. As shown in Figure 7.1, these two scenarios have a larger rectangular area for more turbines and obstacles where turbines can't be installed. Thus these two scenarios are far more challenging than the first two scenarios. The number of turbines for Comp 1 and Comp 3 are fixed at 220 and 710, respectively. The shaded rectangles in Figure 7.1 indicate the position and size of obstacles in scenario Comp 1 and Comp 3, respectively. In these scenarios, the turbines cannot be placed inside the obstacles.

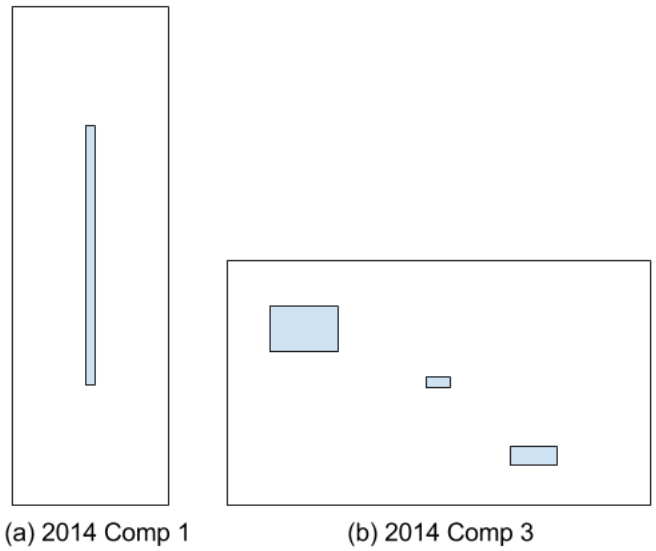

Figure 7.1: Obstacles in scenario Comp 1 and Comp 3. Layouts are not shown to scale.

Figure 7.2 shows the wind speed and wind directions in each scenario [56, 105]. Each wind rose gives the proportionate expected wind speed in each direction. Directions are discretised into 24 sectors. Each concentric circle depicted in the diagram represents a different expected wind speed, starting from zero at the centre to higher expected speed at the outer circles. The length of each colour-coded "spoke" around the circle illustrates the relative wind speed in the pointed direction.

Table Table 7.1 shows the basic descriptions of four scenarios, including dimensions, and the fixed number of blocks (across and down) for use with the BlockCopy operator. In each scenario, the entire site is

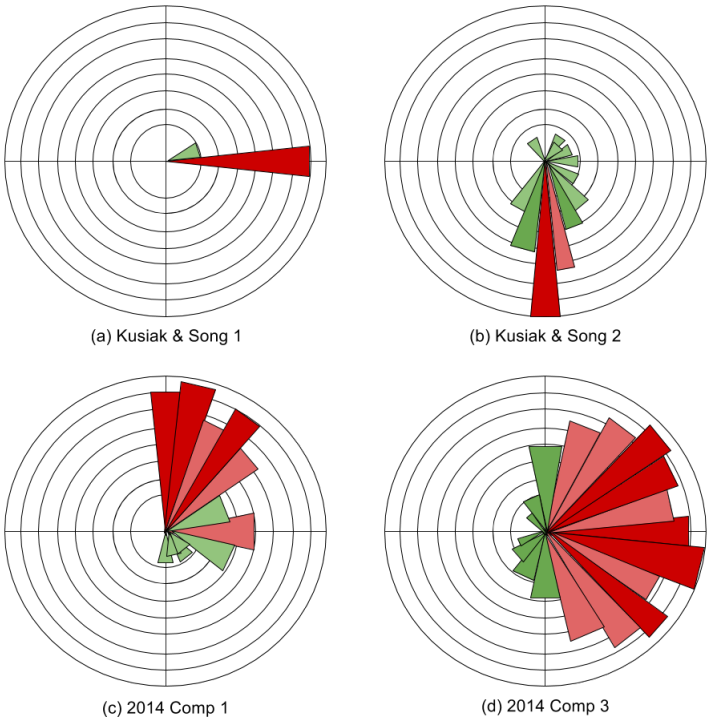

Figure 7.2: Wind rose used in each scenario. These wind rose diagrams are not shown to scale.

divided into approximate 1km by 1km blocks. The Weibull k parameter is also different from each scenario to provide variation in the estimates of wind speed at turbine hub height.

Scenario	Width (km)	Height (km)	# Turbines	Width (blocks)	Height (blocks)	Obstacles?	k
Kusiak & Song 1 [56]	4.0	4.0	100	4	4	No	2.0
Kusiak & Song 1 [56]	4.0	4.0	100	4	4	No	2.0
2014 Comp 1 [105]	3.5	16.1	220	3	16	Yes	2.187-3.624
2014 Comp 3 [105]	15.8	11.3	710	16	11	Yes	2.016-4.473

Table 7.1: Wind Scenario dimensions, number of turbines, number of blocks and k parameter.

7.2 Objective Function

As mentioned before, the simple objective function is accurate enough to test the ability of the optimisation methods. Similar to most of the studies in the literature, this work utilises a simplified wind farm eval-

uation method that makes it more practical to be integrated into optimisation procedures and programs. Because of its simplicity, it is more computationally feasible, and it can provide adequate accuracy for the simulation [73]. However, it still has high time complexity. The wake model used here is only accurate for the far wake effect, and therefore no turbine should be place to near to another turbine. The minimum distance between any two turbines is 8 times of the turbine diameter (see Figure 2.3).

In our work, the objective function used here is an extended version from 2015 Wind Farm Layout Optimisation Competition [105], which was originally proposed by Kusiak and Song [54]. By calculating the total cost of the farm (including construction and yearly operating costs) and dividing that by the total power output of the wind farm, the cost is defined as the expected cost of per kilowatt energy output. The objective function is defined as:

$$cost = \frac{(c_t \times n + c_s \times \lfloor \frac{n}{m} \rfloor)(\frac{2}{3} + \frac{1}{3} \times e^{-0.00174n^2}) + C_{O\&M} \times n}{(1-(1+r)^{-y}) \Big/ n} \times \frac{1}{8760 \times P} + \frac{0.1}{n} \tag{7.1}$$

where $c_t = 750,000$ is the cost of a turbine in USD; $c_s = 8,000,000$ is the cost of subsection in USD; $m = 30$ is the number of turbines per subsection; $r = 0.3$ is the interest rate; $y = 20$ is the lifetime of the farm in years; $C_{O\&M} = 20,000$ is the cost of operations and maintenance in USD; n is the number of turbines; and P is the total energy output of the farm. This objective function is simplified, for example, the runtime (3.4 GHz Intel Core i5) for one evaluation using this objective function on a wind farm scenario with 100 WTs and 720 WTs are approximately 410 milliseconds and 8200 milliseconds, respectively. Even this objective function is simplified, it is still adequately accurate for the evaluation [26].

7.3 Evaluation Setup

In this thesis, the basic evolutionary algorithm is (μ, λ)-ES (elitism = 1). By elitism = 1, it is meant that the global best layout found so far will always randomly replace one offspring in the next generation. We employed the MLP (-H 20) neural network and the M5P (-M 5) tree-based regression model as the surrogate models in the surrogate-assisted (μ, λ)-ES (elitism = 1). We used Pre-selection and Best surrogate manage strategies. Each algorithm only employs one of these two management strategies.

As shown in Table 7.3, there are 10 different algorithms using various combinations of different surrogate models, different surrogate management strategies and different BlockCopy operators. We evaluated these 10 algorithms with three sets of population configurations including $(6, 12)$, $(10, 20)$ and $(20, 40)$. During the initialisation of optimisation, the candidate wind farm layouts are randomly generated. Please note that due to limited space, we used abbreviations to represent the names of different algorithms.

Algorithm Parameter	Setting & Value
Number of evaluation using the real fitness function	2,000
Number of approximation using the surrogate model	Unlimited
Size of the training data	1,000
How many times of surrogate model training and re-training	5
Surrogate model initial construction after real evaluation	1,000
Surrogate re-training after real evaluations	1200, 1400, 1600, 1800

Table 7.2: The discreption of different evolutionary algorithms.

As shown in Table 7.2, all 10 algorithms are limited to use 2,000 real fitness evaluations in total. The standard two evolutionary algorithms terminate when they have called the real fitness function 2,000 times. The surrogate-assisted algorithms use the real fitness function for the first 1,000 real fitness evaluations. Meanwhile, the training data is collected and the surrogate model is constructed. Then the surrogate models are employed in the evolutionary algorithms for fitness approximation along with the last 1,000 real fitness evaluations (see Figures 5.5

and 5.6).

There are four wind farm scenarios for evaluation (see Section 7.1). The optimisation process can use unlimited surrogate evaluations but terminates after 2,000 real fitness evaluations. Each algorithm only employs one of the BlockCopy Mutate operator or the BlockCopy Crossover operator to generate offspring. Thus there is no run using both of them.

Each algorithm was repeated 30 times on each scenario. There are total of 10 algorithms $\times$ 3 population configurations $\times$ 4 scenarios $\times$ 30 repetitions = 3,600 experimental runs.

Abbreviation of Algorithm	Description
ES.C	(μ, λ)-evolutionary strategy using BlockCopy Crossover operator without surrogate models
ES.M	(μ, λ)-evolutionary strategy using BlockCopy Mutate operator without surrogate models
MLP.Best.C	(μ, λ)-evolutionary strategy with MLP surrogate model managed by Best surrogate management strategy using BlockCopy Crossover operator
MLP.Best.M	(μ, λ)-evolutionary strategy with MLP surrogate model managed by Best surrogate management strategy using BlockCopy Mutate operator
MLP.Pre.C	(μ, λ)-evolutionary strategy with MLP surrogate model managed by Pre-selection surrogate management strategy using BlockCopy Crossover operator
MLP.Pre.M	(μ, λ)-evolutionary strategy with MLP surrogate model managed by Pre-selection surrogate management Strategy using BlockCopy Mutate operator
M5P.Best.C	(μ, λ)-evolutionary strategy with M5P surrogate model managed by Best surrogate management strategy using BlockCopy Crossover operator
M5P.Best.M	(μ, λ)-evolutionary strategy with M5P surrogate model managed by Best surrogate management strategy using BlockCopy Mutate operator
M5P.Pre.C	(μ, λ)-evolutionary strategy with M5P surrogate model managed by Pre-selection surrogate management strategy using BlockCopy Crossover operator
M5P.Pre.M	(μ, λ)-evolutionary strategy with M5P surrogate model managed by Pre-selection surrogate management strategy using BlockCopy Mutate operator

Table 7.3: The discreption of different evolutionary algorithms.

7.4 Evaluation Results

In this section we show the distribution of best fitness (lowest cost of energy) achieved by each evolutionary algorithm (see Table 7.3) with different population configurations on four benchmark wind farm scenarios (see Table 7.1). The convergence history for the best layouts found by each algorithm with different population configurations are also depicted.

7.4.1 Wind Farm Scenario Kusiak & Song 1 [54]

The box-and-whisker plots illustrating the distribution of best fitness (lowest cost of energy) achieved by each algorithm (see Table 7.3) with different population configurations on the Kusiak & Song 1 [54] wind farm scenario are shown in Figures 7.3 to 7.5. On the y-axis we show the fitness (cost of energy) whereas the names of different evolutionary algorithms are shown on the x-axis.

The MLP-assisted EAs and M5P-assisted EAs are not ideal in comparison to the EAs without any surrogate model. We can observe that the (μ, λ)-EA using BlockCopy Mutate operator without surrogate models obtained the best wind farm layout with fitness value (lowest cost of energy).

On the other hand, regardless of the surrogate models and surrogate management strategies, the BlockCopy Mutate operator has a better chance for finding better solutions compared to the BlockCopy Crossover operator. Among the eight surrogate-assisted evolutionary algorithms, surrogate-assisted algorithms using Best surrogate management strategy are more likely to obtain better final solutions. The MLP surrogate-assisted $(10, 20)$-EA using BlockCopy Mutate operator and Best surrogate management strategy obtained better wind farm layouts than other surrogate-assisted EAs.

Different population configurations showed significant differences. The $(10, 20)$-EA (Elitism = 1) achieved better fitness values compared to $(6, 12)$-EA (Elitism = 1) and $(20, 40)$-EA (Elitism = 1). Under $(10, 20)$ and $(20, 40)$ population configurations, the EAs using BlockCopy Mutate operator, MLP surrogate model and Best strategy outperformed the rest of the surrogate-assisted algorithms.

It is clear that under $(6, 12)$ population configuration, the M5P surrogate-assisted EAs achieved better results compared to the MLP surrogate-assisted EAs. However, when the population size is increased to $(10, 20)$ and $(20, 40)$, the M5P surrogate-assisted EAs are not as good as the EAs

using MLP surrogate models.

Bigger population size also results in worse performance using surrogate models. Solutions obtained under surrogate-assisted $(6, 12)$-EAs are statistically better than the results obtained the same algorithms using $(10, 20)$ and $(20, 40)$ population configuration.

As shown in Figure 7.8, an interesting case is the best finding obtained by $(6, 12)$-EA using BlockCopy Mutate operator. The fitness value of the global best layout at the beginning of optimisation is clearly the best (lowest cost of energy). As a consequence, at then end of 2,000 real evaluations, the obtained final solution is obviously the winner compared to other algorithms.

The convergence history of best layouts found by each algorithm (see Table 7.3) using different population configurations on the Kusiak & Song 1 [54] wind farm scenario are depicted in Figures 7.6 to 7.8. On the y-axis we show the fitness (cost of energy) of the wind farm layout after x real fitness evaluations.

It can be seen that the convergence speed of surrogate-assisted algorithms using smaller population size is similar to the EAs without surrogate models. The larger population size considerably slows down the convergence speed.

Table 7.4 illustrates the average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario Kusiak & Song 1 [54].

Algorithm	ES.C	ES.M	MLP.Best.C	MLP.Best.M	MLP.Pre.C	MLP.Pre.M	M5P.Best.C	M5P.Best.M	M5P.Pre.C	M5P.Pre.M
(6, 12)-EA	821285.4	819707.1	1713122	1675798	1433692	1520438.8	836327.3	834299.6	817653.6	824259.8
(10, 20)-EA	755233.7	753360	2190837.8	2199496.2	591829.8	597184.9	907275.8	912280.6	1026685.7	1028500.2
(20,40)-EA	630947.1	629554.9	3123168.1	3104373.9	1192593.8	1341508.8	963695.9	969615.8	1074110.3	1084529.3

Table 7.4: The average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario Kusiak & Song 1 [54]. The unit is milliseconds.

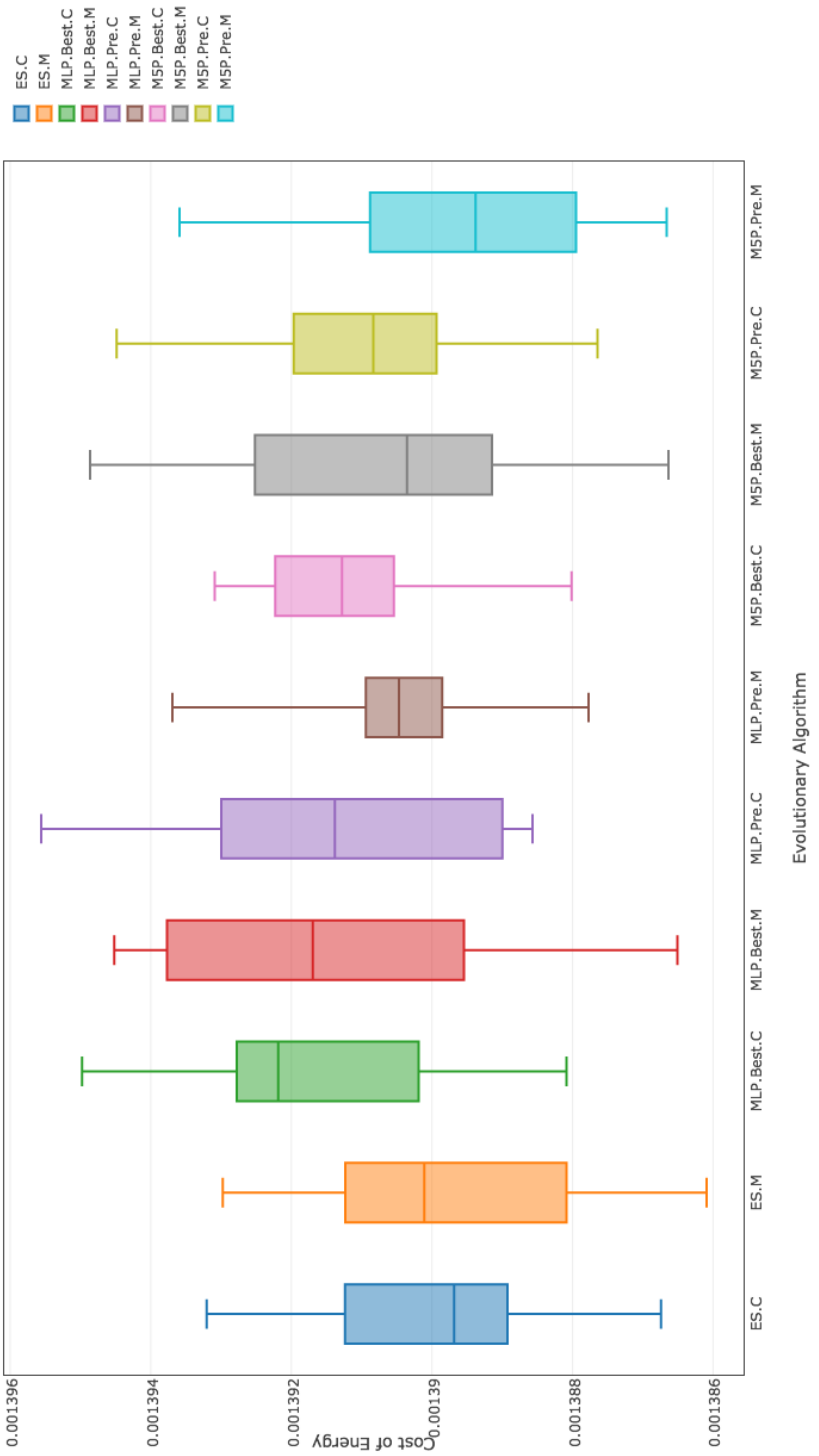

Figure 7.3: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario.

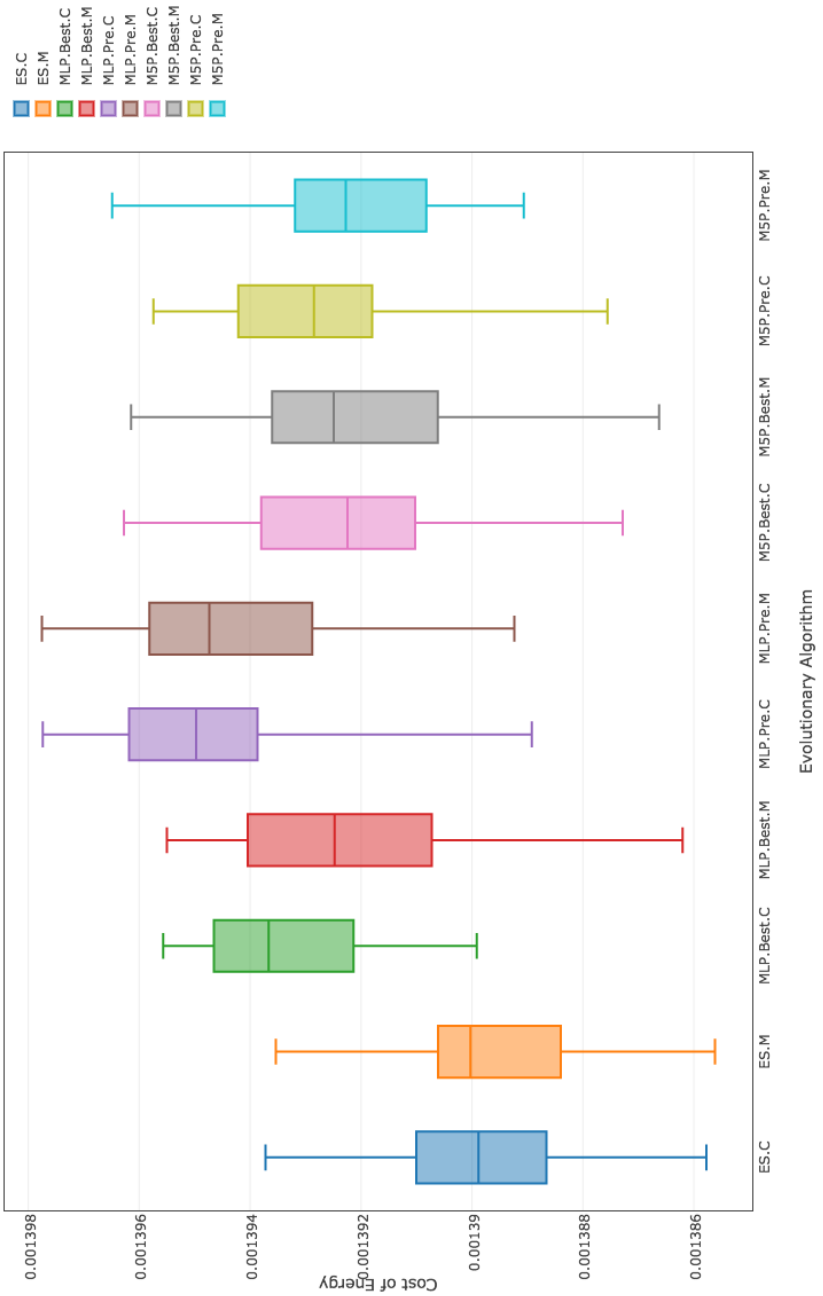

Figure 7.4: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario.

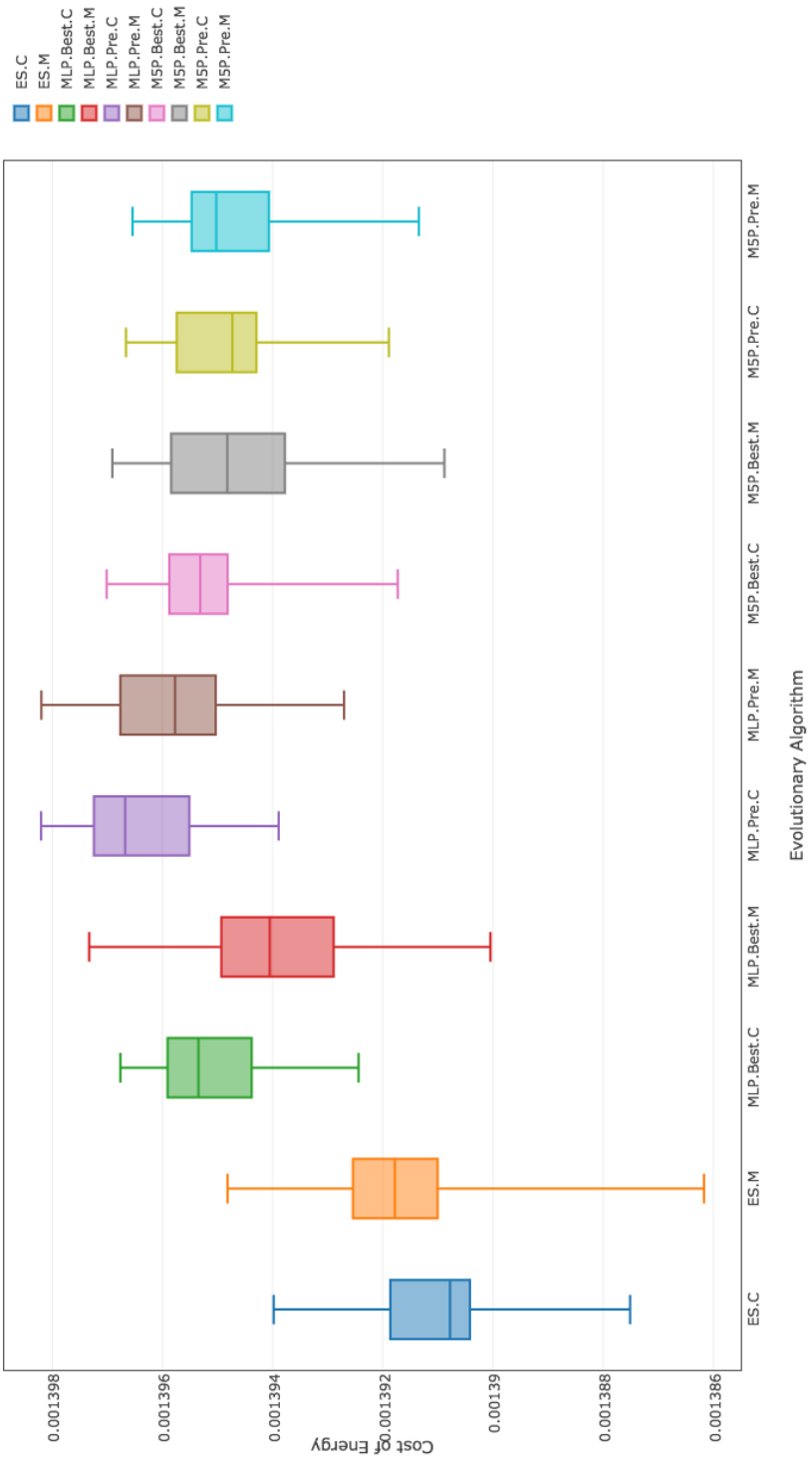

Figure 7.5: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario.

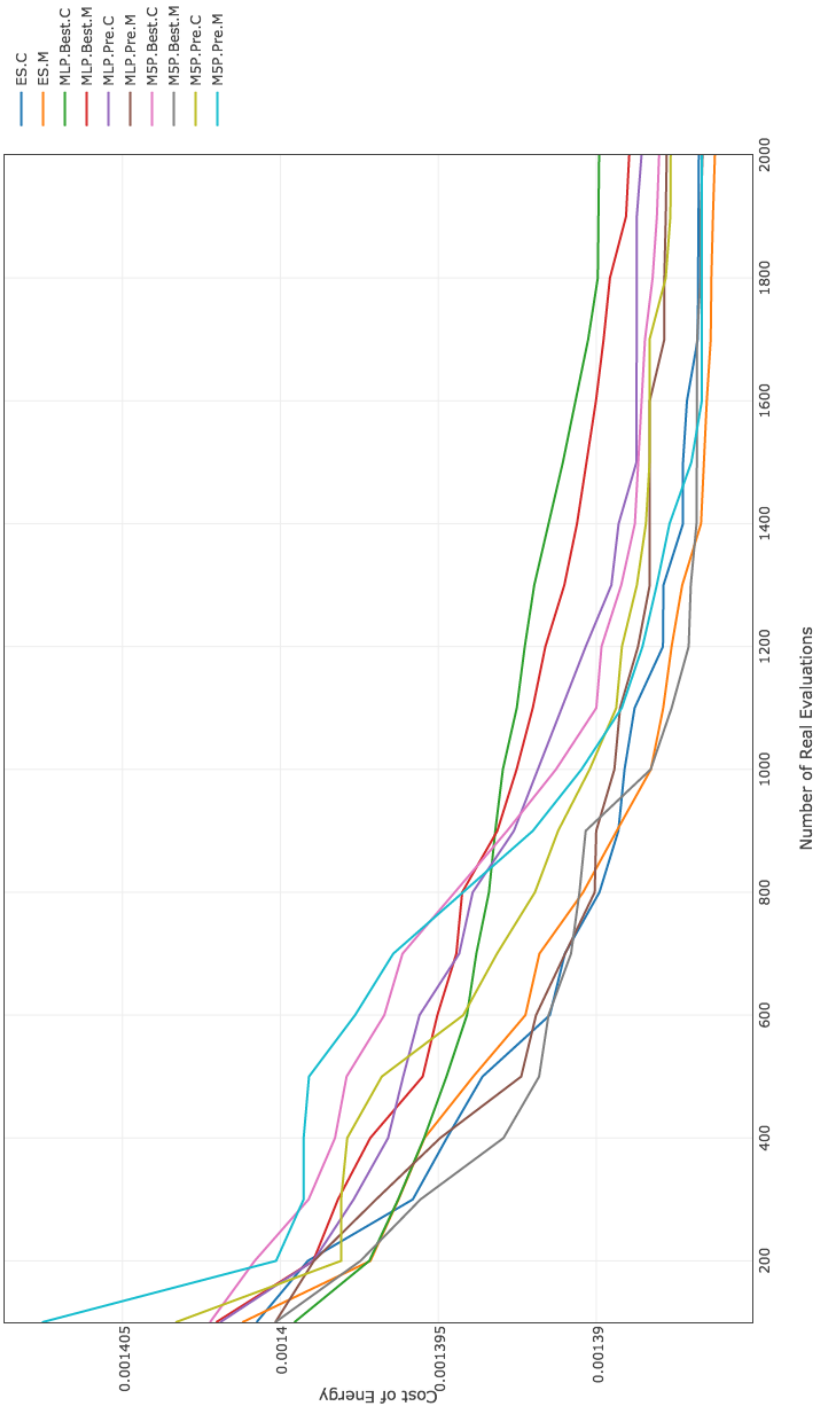

Figure 7.6: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario.

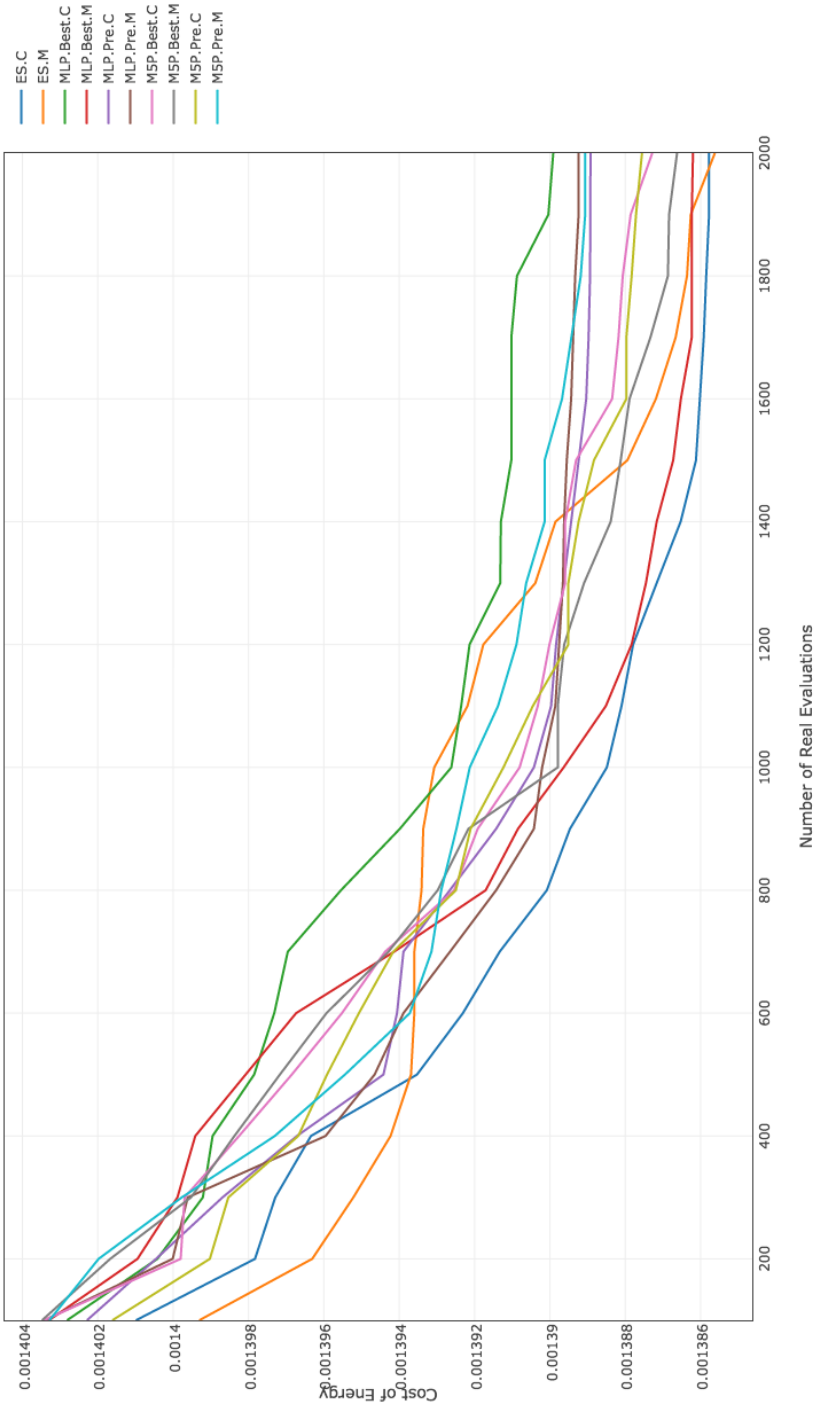

Figure 7.7: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario.

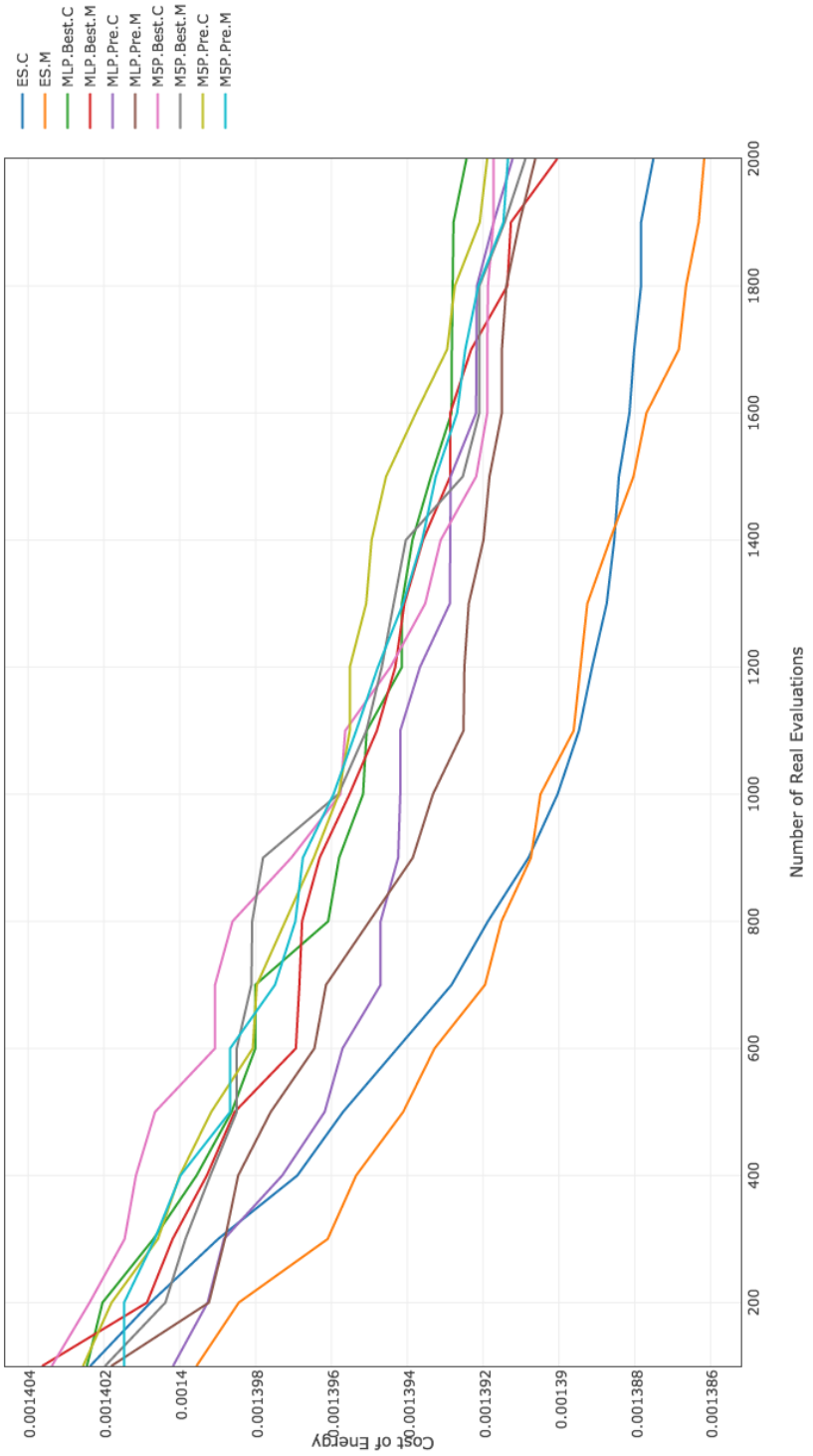

Figure 7.8: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the Kusiak & Song 1 [54] wind farm scenario.

7.4.2 Wind Farm Scenario Kusiak & Song 2 [54]

The box-and-whisker plots illustrating the distribution of best fitness (lowest cost of energy) obtained by each algorithm (see Table 7.3) with different population configurations on the Kusiak & Song 2 [54] wind farm scenario are illustrated in Figures 7.9 to 7.11. The names of different evolutionary algorithms are depicted on the x-axis whereas we show the fitness (cost of energy) on the y-axis.

Overall, the $(10, 20)$-ES using BlockCopy Crossover operator obtained the best solution (lowest cost of energy). The $(20, 40)$-ES using BlockCopy Crossover operator outperformed the $(20, 40)$-ES using BlockCopy Mutate operator whereas the $(6, 12)$-ES using BlockCopy Crossover operator obtained worse solution compared to the $(6, 12)$-ES using BlockCopy Mutate operator.

One interesting case is that the MLP-assisted EA using Pre-selection strategy and BlockCopy Crossover operator achieved the lowest cost of energy under $(6, 12)$ population configuration. Another interesting case is that the best findings obtained by MLP-assisted $(6, 12)$-EA using Pre-selection strategy with BlockCopy Mutate operator and M5P-assisted $(6, 12)$-EA using Best strategy with BlockCopy Mutate operator are both slightly better than the best layout found by the $(6, 12)$-ES using BlockCopy Crossover operator. However, these two cases are not general phenomenons and may be due to luck.

Similar to wind farm scenario Kusiak & Song 1 [54], when the population size is increased to $(10, 20)$ and $(20, 40)$, the surrogate-assisted EAs obtained worse solutions compared to the EAs without surrogate models.

Interestingly, under $(10, 20)$ population configuration, the M5P-assisted EA using Best strategy and BlockCopy Crossover operator achieved the lowest cost of energy compared to MLP-assisted EAs. Similarly, the M5P-assisted $(20, 40)$ EA using Best strategy and BlockCopy Mutate operator achieved the lowest cost of energy compared to MLP-assisted EAs.

The convergence history of best layouts found by each algorithm (see Table 7.3) using different population configurations on the Kusiak & Song 2 [54] wind farm scenario are depicted in Figures 7.12 to 7.14. On the y-axis we show the fitness (cost of energy) of the wind farm layout after x real fitness evaluations.

Similar to the the Kusiak & Song 1 [54] wind farm, smaller population size showed faster convergence trend. Interestingly, as can been in Figure 7.14, at the end of optimisation process (i.e. after 1800 real evaluations), several surrogate-assisted EAs showed obvious fitness improvement.

Table 7.5 shows the average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario Kusiak & Song 2 [54].

Algorithm	ES.C	ES.M	MLP.Best.C	MLP.Best.M	MLP.Pre.C	MLP.Pre.M	M5P.Best.C	M5P.Best.M	M5P.Pre.C	M5P.Pre.M
(6, 12)-EA	824369.8	839767	2011191	2110340	2183347	2180243	1663695	1780589	778416	792370
(10, 20)-EA	744369.8	739767	2484757.3	2806473	2694619.6	2710504	1900238	1817011	1055657.8	1062524.6
(20,40)-EA	644361.2	630497.2	3101704	3398646	3012538	3256776	1098265	1125650	1078723	1086093

Table 7.5: The average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario Kusiak & Song 2 [54]. The unit is milliseconds.

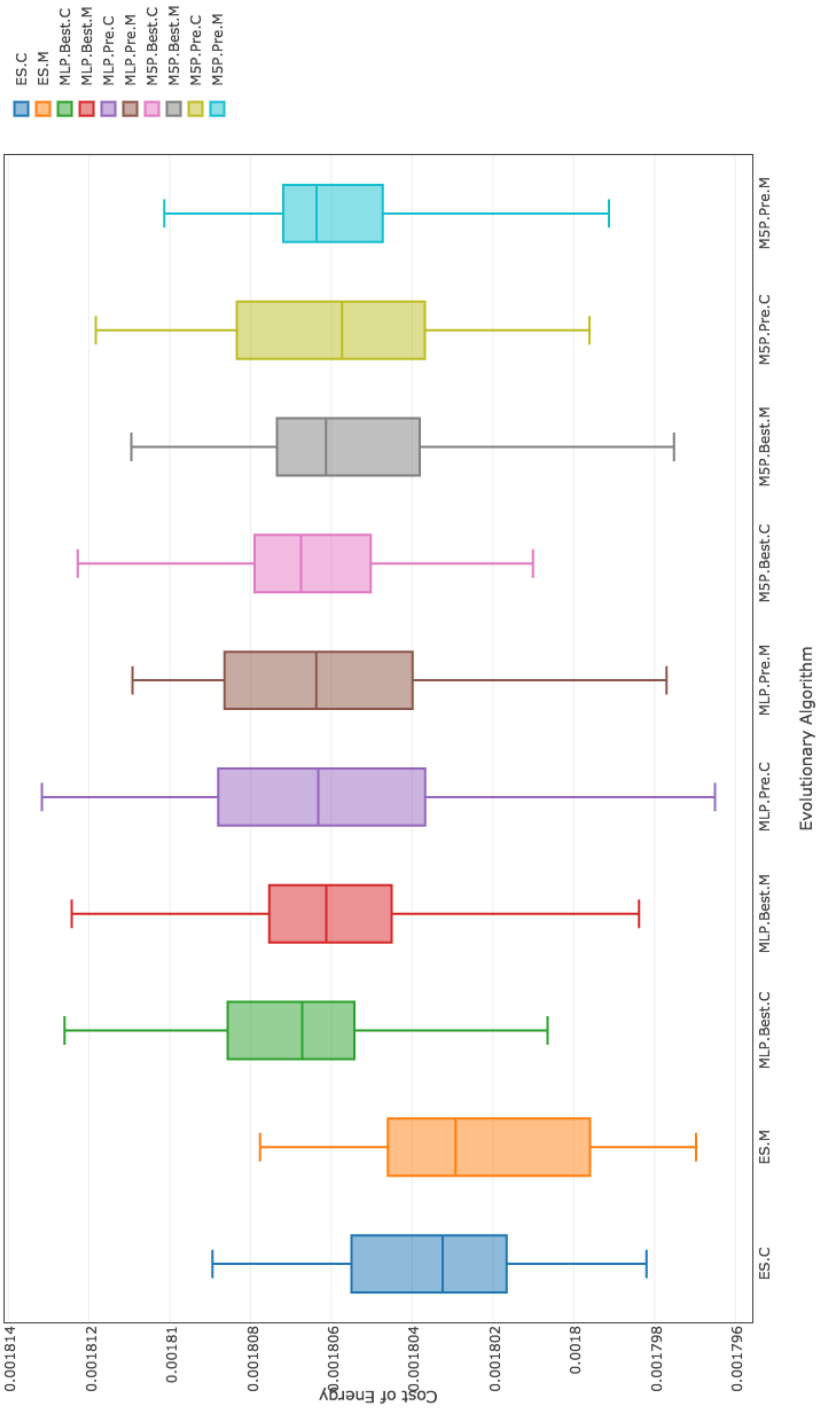

Figure 7.9: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario.

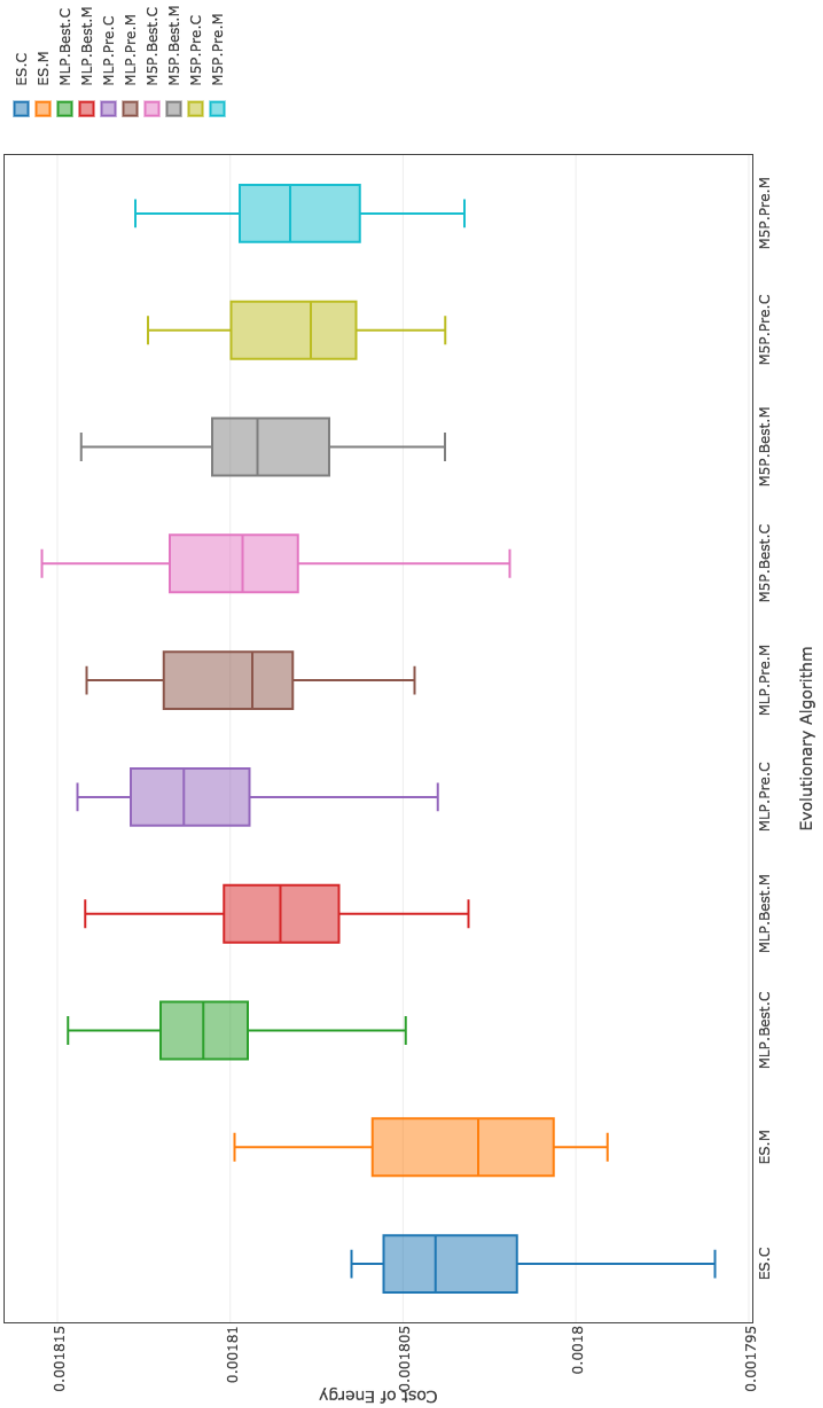

Figure 7.10: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario.

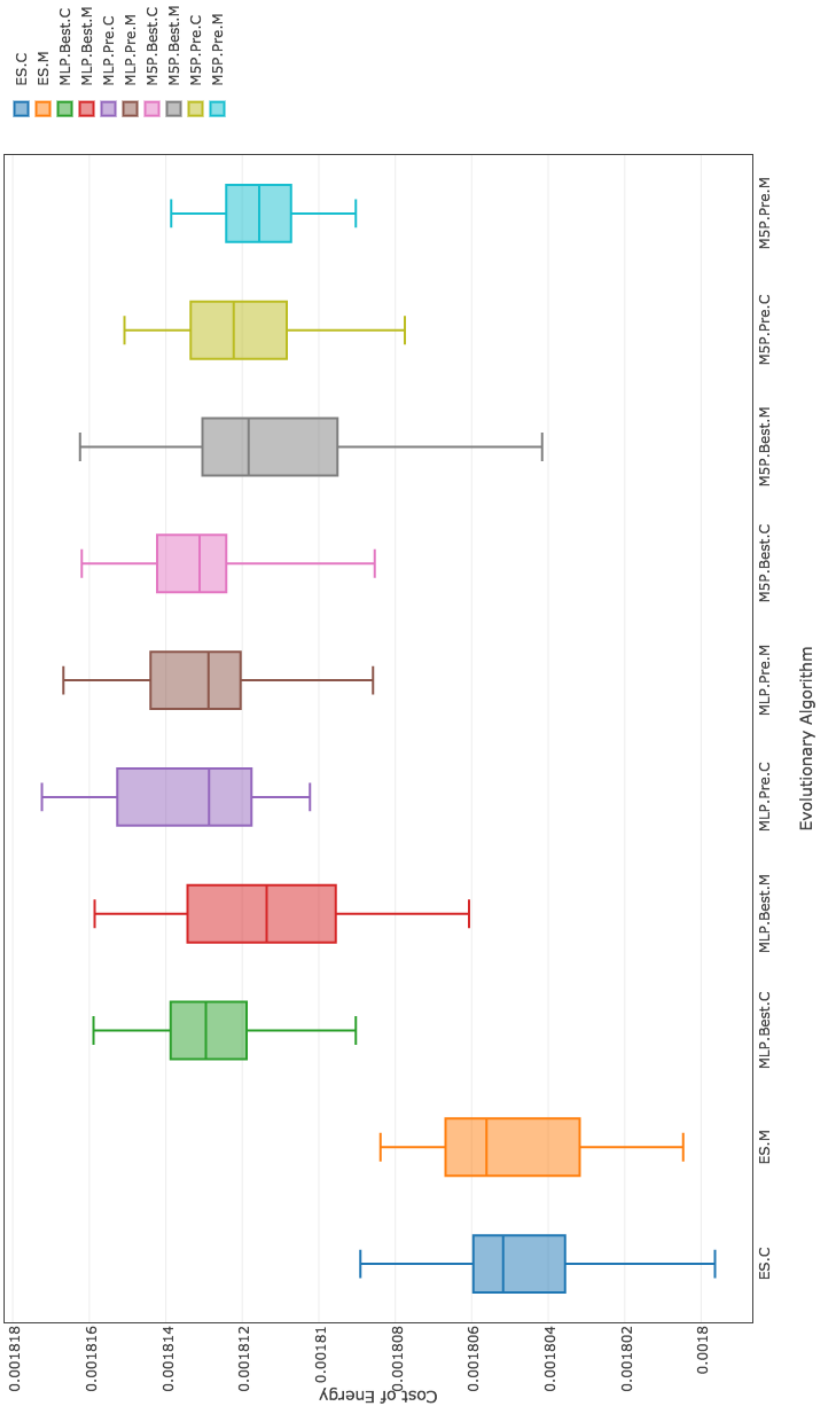

Figure 7.11: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario.

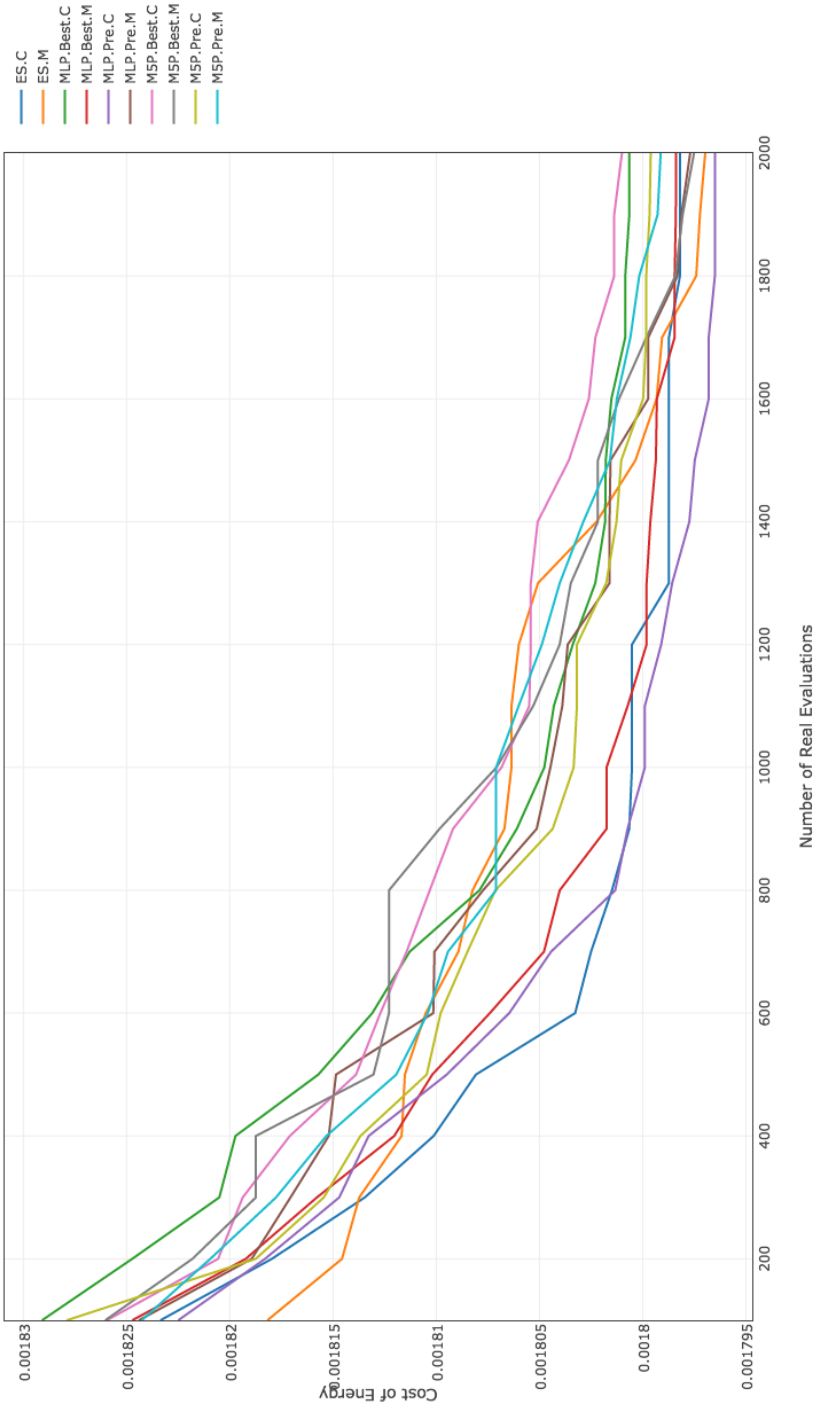

Figure 7.12: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario.

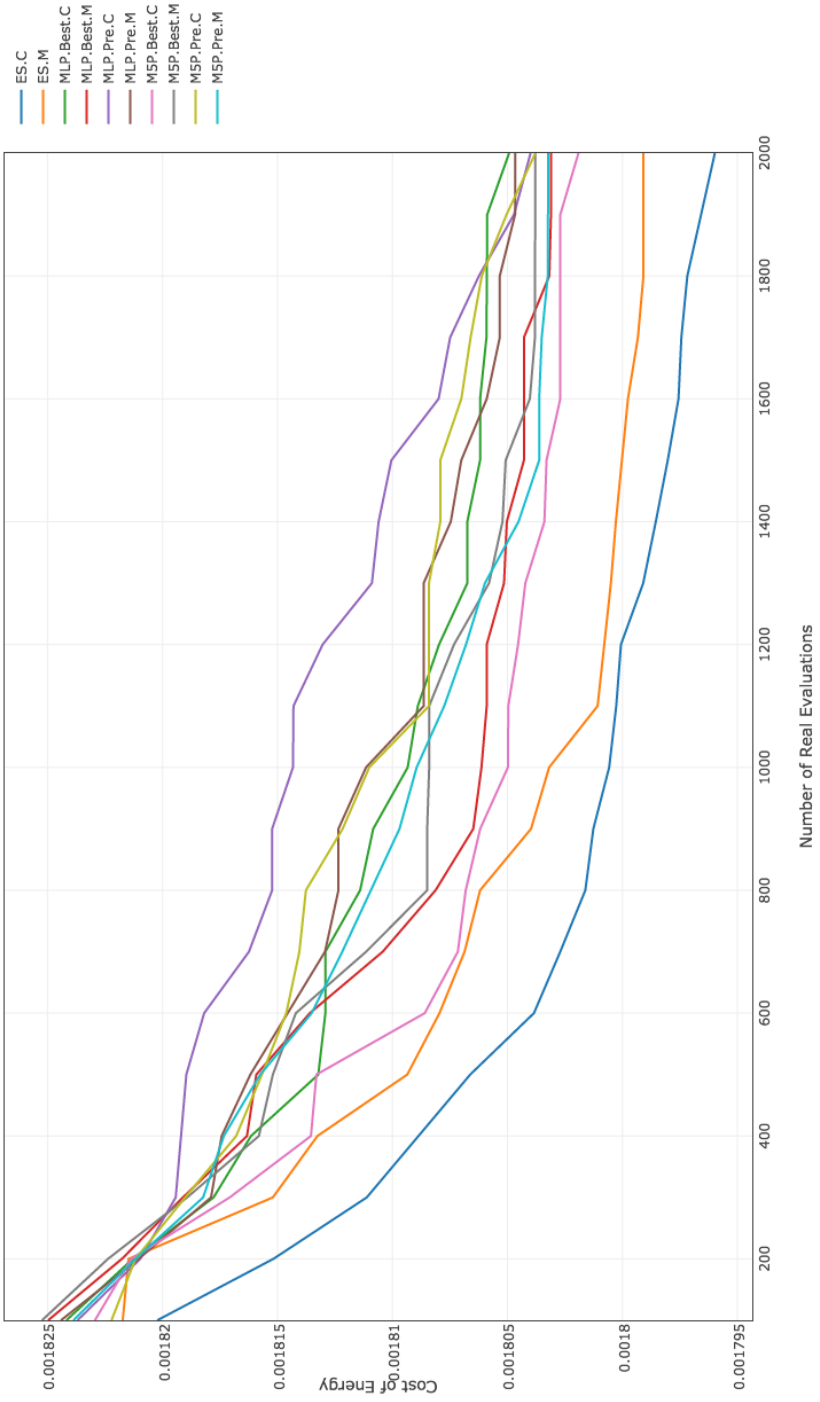

Figure 7.13: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario.

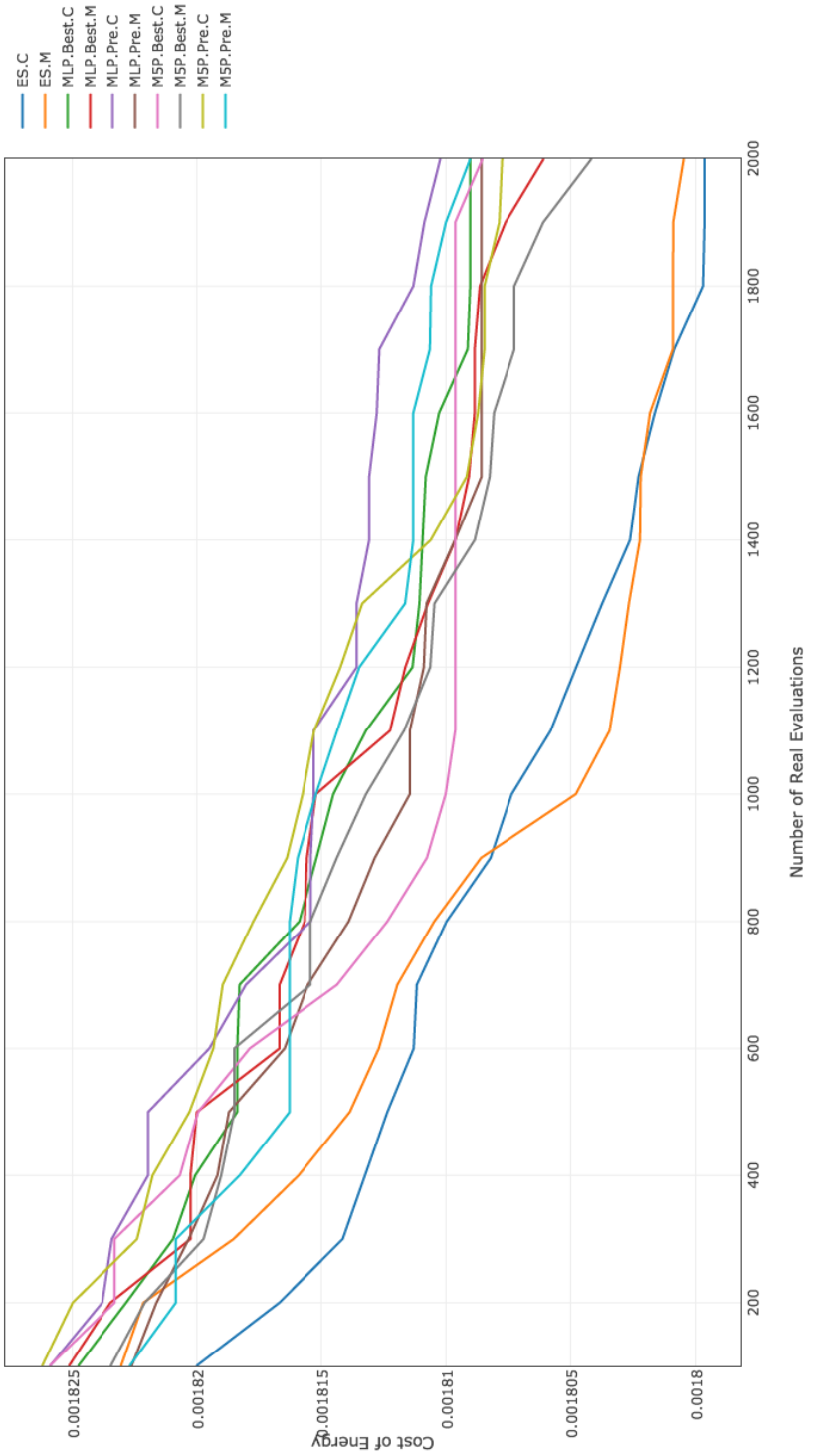

Figure 7.14: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the Kusiak & Song 2 [54] wind farm scenario.

7.4.3 Wind Farm Scenario 2014 Comp 1 [105]

The box-and-whisker plots showing the distribution of best fitness (lowest cost of energy) achieved by each algorithm (see Table 7.3) with different population configurations on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario are depicted in Figures 7.15 to 7.17. On the y-axis we illustrate the fitness (cost of energy) whereas the names of different evolutionary algorithms are shown on the x-axis.

Similar to the evaluation results on the wind farm scenario Kusiak & Song 1 and 2 [54], the overall performance of MLP-assisted EAs and M5P-assisted EAs are not ideal in comparison to the EAs without any surrogate model. Among all the experimental results the $(6, 12)$-EA using BlockCopy Crossover operator without surrogate models obtained the best wind farm layout (lowest cost of energy).

Comparing to wind farm scenario Kusiak & Song 1 and 2 [54], the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario is more difficult since it consists of 220 wind turbines. The results obtained by surrogate-assisted EAs are significantly worse than the EAs without any surrogate models.

In Figure 7.15, the M5P-assisted EA using BlockCopy Mutate operator with Best strategy achieved better final solution compared to other surrogate-assisted EAs. But it is still not as good as the EAs without surrogate models. In contrast to that, as shown in Figure 7.16, the MLP-assisted EA using BlockCopy Mutate operator with Best strategy achieved better final solution compared to other surrogate-assisted EAs under $(10, 20)$ population configuration. Despite the different population configurations, the M5P-assisted EAs performed slightly better that the MLP-assisted EAs.

The convergence history of best layouts found by each algorithm (see Table 7.3) using different population configurations on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario are depicted in Figures 7.18 to 7.20. On the y-axis we show the fitness (cost of energy)

of the wind farm layout after x real fitness evaluations.

It is clear that the convergence speed of each EA without surrogate models is relatively faster compared to the surrogate-assisted EAs. As the scenario becames more complicated, the larger population size results in much slower convergence process.

Table 7.6 shows the average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario 2014 Wind Farm Layout Optimisation Comp 1 [105].

Algorithm	ES.C	ES.M	MLP.Best.C	MLP.Best.M	MLP.Pre.C	MLP.Pre.M	M5P.Best.C	M5P.Best.M	M5P.Pre.C	M5P.Pre.M
(6, 12)-EA	4925564	4977093	6569461	7428532.3	4164281	4227197	4095743	4286463	3264965	3201294
(10, 20)-EA	3363361.9	3397792	7073659	7737979	4667536.3	4779960.7	2844669	2917213	2844669	2917213
(20,40)-EA	986271.3	985273.7	7132351	8078473	6569430	6717710	2252649.3	2224593	2802772	2864939

Table 7.6: The average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario 2014 Wind Farm Layout Optimisation Comp 1 [105]. The unit is milliseconds.

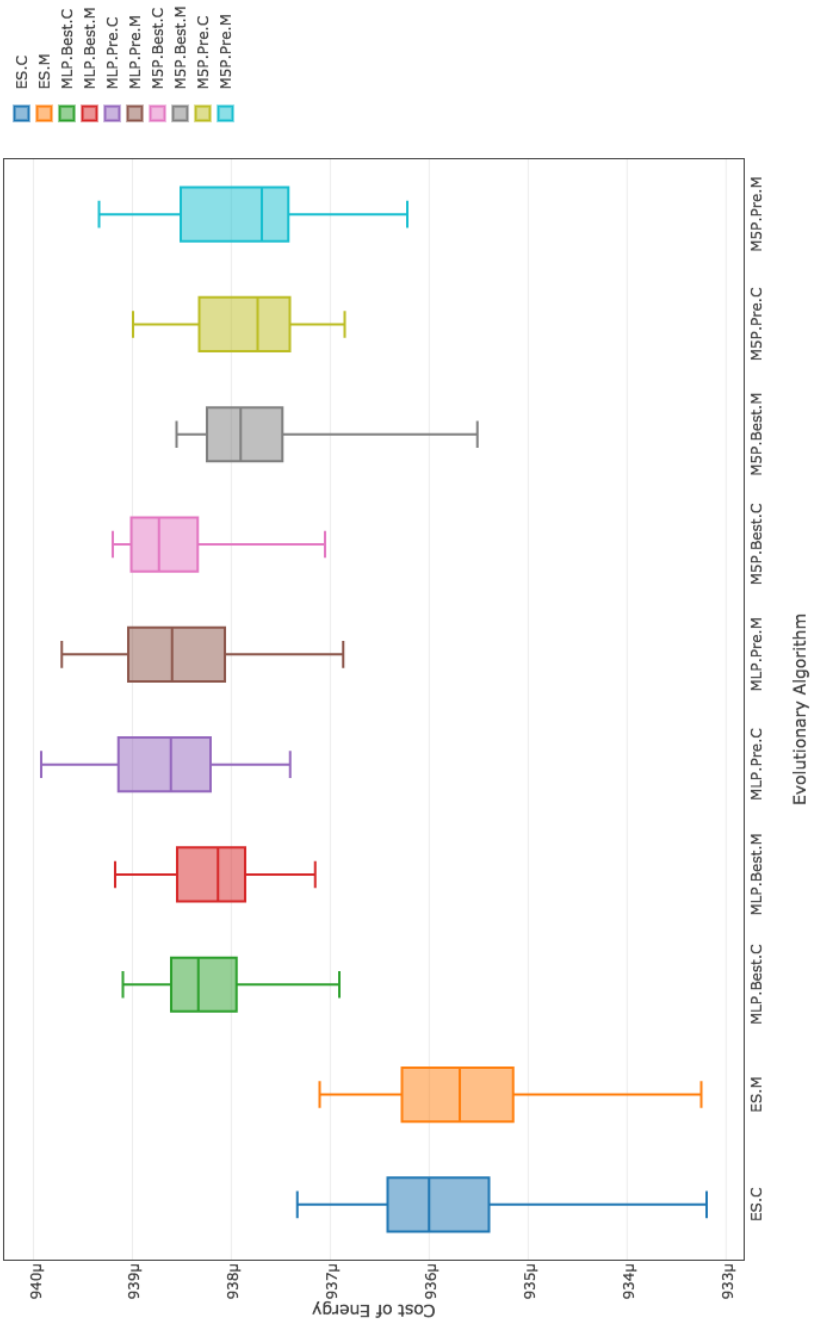

Figure 7.15: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$.

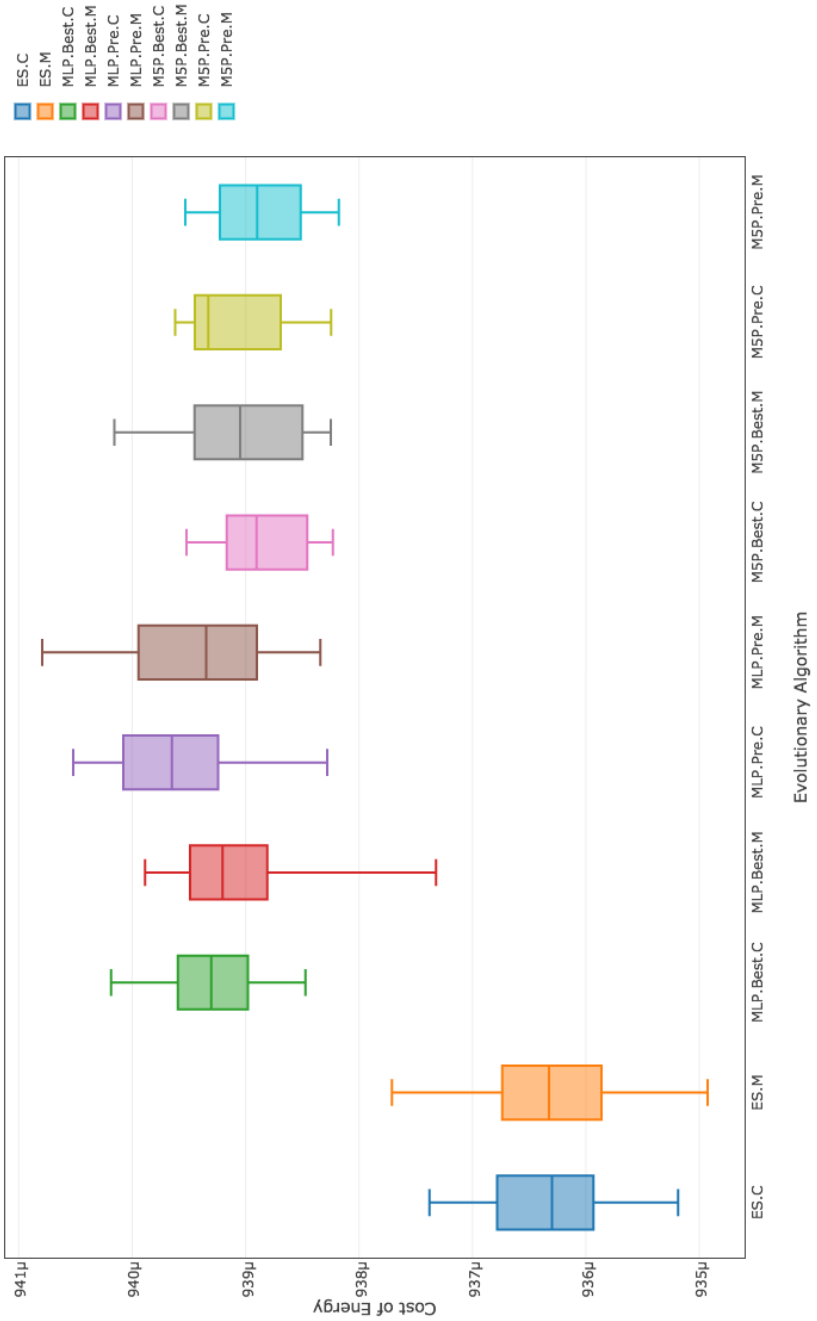

Figure 7.16: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$.

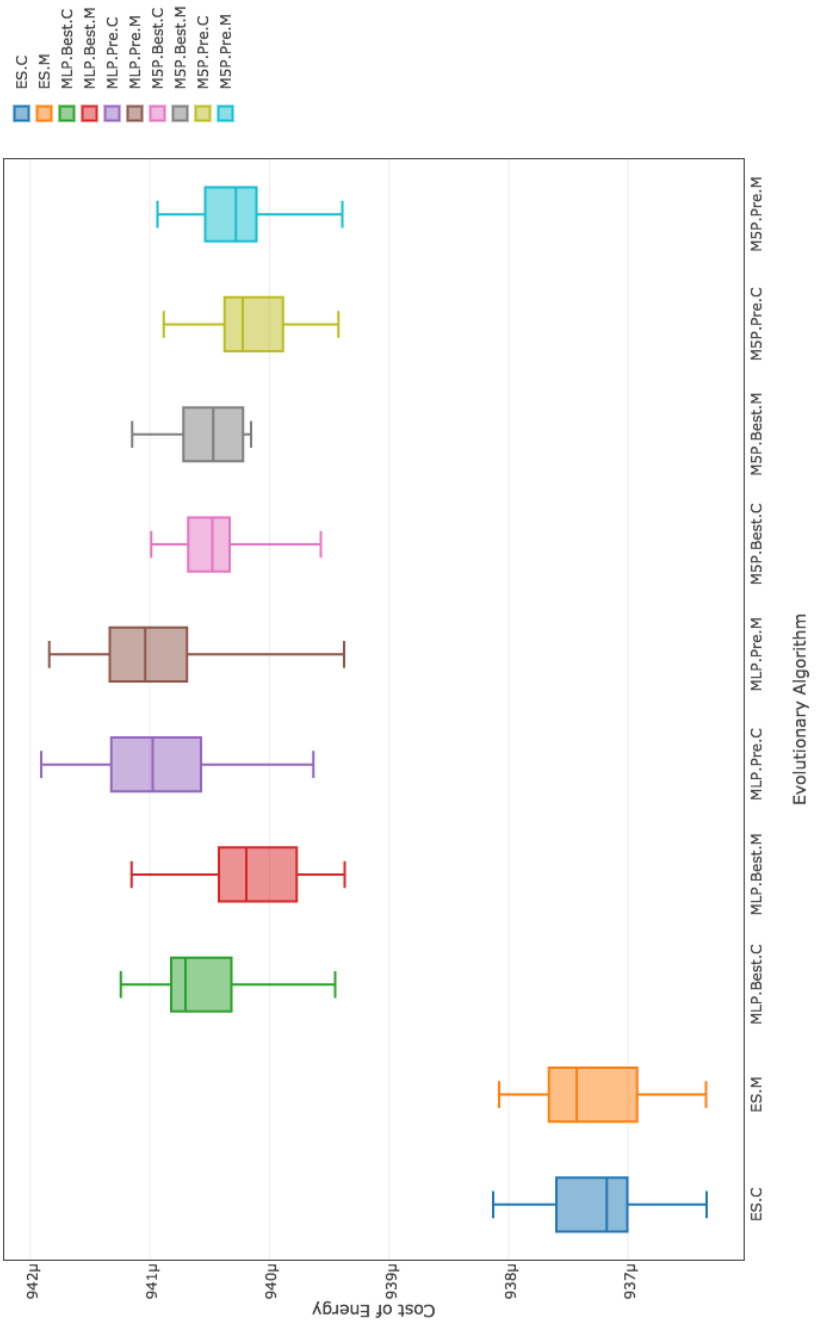

Figure 7.17: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$.

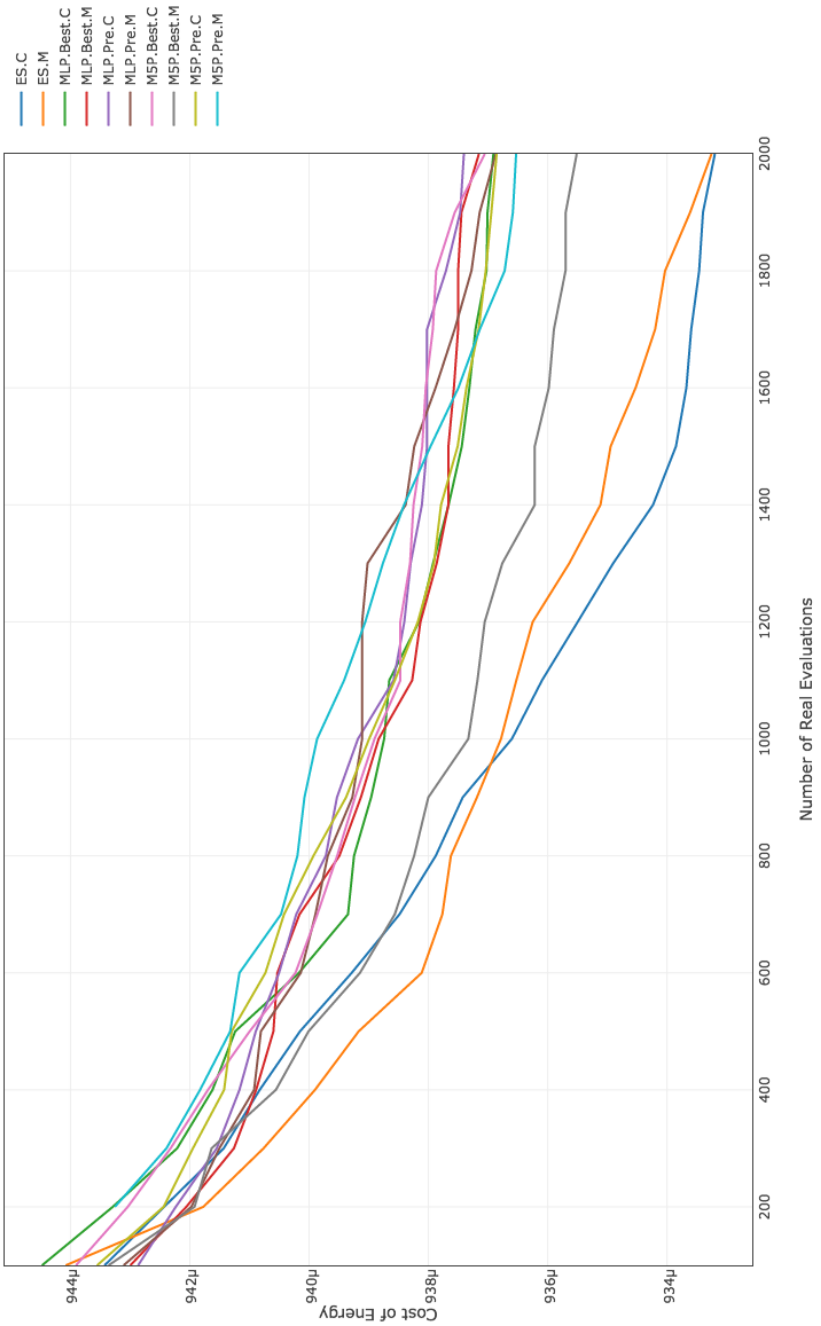

Figure 7.18: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$.

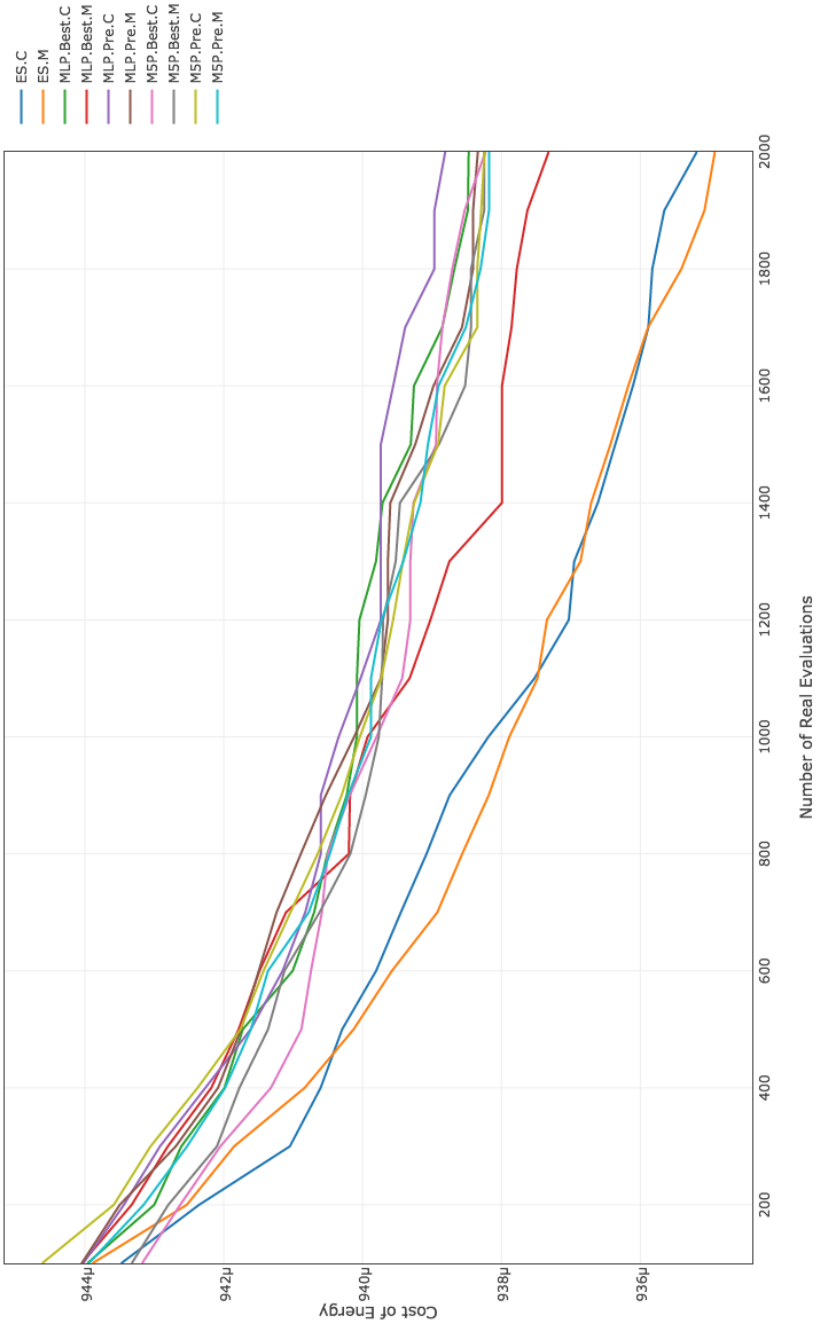

Figure 7.19: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$.

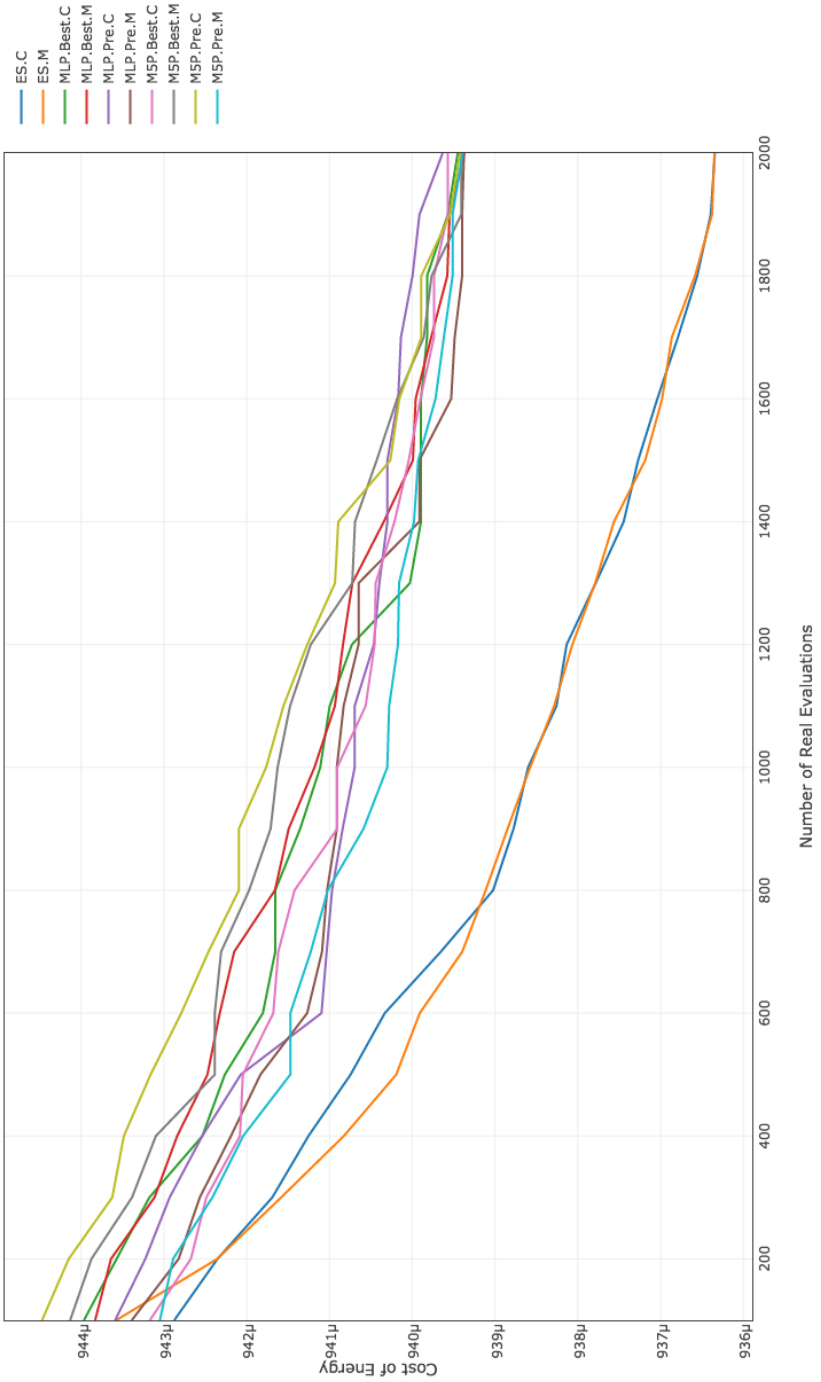

Figure 7.20: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 1 [105] wind farm scenario. Please note that $\mu = 10^{-6}$.

7.4.4 Wind Farm Scenario 2014 Comp 3 [105]

The box-and-whisker plots showing the distribution of best fitness (lowest cost of energy) achieved by each algorithm (see Table 7.3) with different population configurations on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario are shown in Figures 7.21 to 7.23. The names of different evolutionary algorithms are shown on the x-axis whereas we depict the fitness (cost of energy) on the y-axis.

As can be seen, the overall performance of MLP-assisted EAs and M5P-assisted EAs are far from ideal in comparison to the EAs without any surrogate model. Among all the experimental results the $(6, 12)$-EA using BlockCopy Crossover operator without surrogate models obtained the best wind farm layout (lowest cost of energy).

In Figure 7.21, the M5P-assisted $(6, 12)$-EA using BlockCopy Mutate operator and Pre-selection strategy showed better performance compared to the other eight surrogate-assisted evolutionary algorithms. In Figure 7.23, the M5P-assisted $(20, 40)$-EA using BlockCopy Crossover operator and Best strategy obtained the best solution among other eight surrogate-assisted evolutionary algorithms.

The 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario has 1420 dimensions. The surrogate-assisted EAs are obviously worse than the EAs without any surrogate model. Similarly, EAs using larger population size also resulted in worse convergence.

As shown in Figure 7.24, another interesting case is the best finding obtained by $(6, 12)$-EA using BlockCopy Mutate operator. The fitness value of the global best layout at the beginning of optimisation is clearly the best (lowest cost of energy). As a result, the final solution obtained at then end of 2,000 real evaluations is obviously the winner compared to other algorithms. The best finding obtained by $(20, 40)$-EA using BlockCopy Crossover operator showed similar trend in Figure 7.26.

The convergence history of best layouts found by each algorithm (see

Table 7.3) using different population configurations on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario are illustrated in Figures 7.24 to 7.26. On the y-axis we show the fitness (cost of energy) of the wind farm layout after x real fitness evaluations.

We can observe that the convergence speed of EAs and surrogate-assisted EAs are significantly different on the most difficult scenario Comp 3. Despite the different population configurations, the overall convergence trend of the surrogate-assisted EAs are quite slow. Similarly, the smaller population size resulted in higher convergence speed for the surrogate-assisted EAs whereas the bigger population size caused more negative influences.

Table 7.7 presents the average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario 2014 Wind Farm Layout Optimisation Comp 3 [105].

Algorithm	ES.C	ES.M	MLP.Best.C	MLP.Best.M	MLP.Pre.C	MLP.Pre.M	M5P.Best.C	M5P.Best.M	M5P.Pre.C	M5P.Pre.M
(6, 12)-EA	14797557	14981138	34290582.3	35189190	28093994	28391726	21561210	20858347	22917809.7	21494890
(10, 20)-EA	14653686.6	14894681	28750866	29417212	28200480.3	28593471.1	15534739	15574711	15518550	15544955
(20,40)-EA	14582412	14586964	26551143	27480130	30188675.5	29806565	22512093	22551998	12984553.7	13088248

Table 7.7: The average elapsed time of thirty runs of each algorithm under three different population configurations for wind farm scenario 2014 Wind Farm Layout Optimisation Comp 3 [105]. The unit is milliseconds.

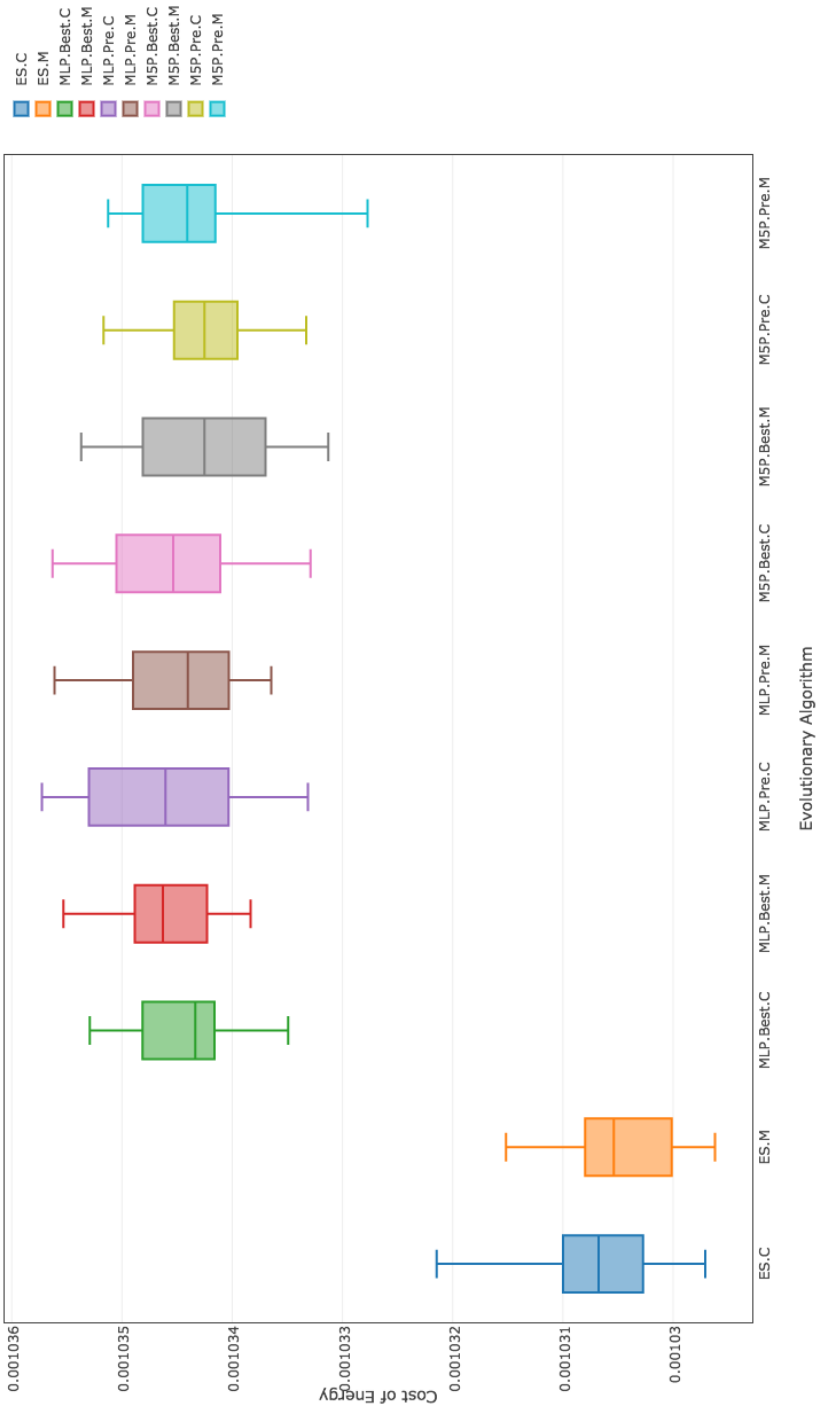

Figure 7.21: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario.

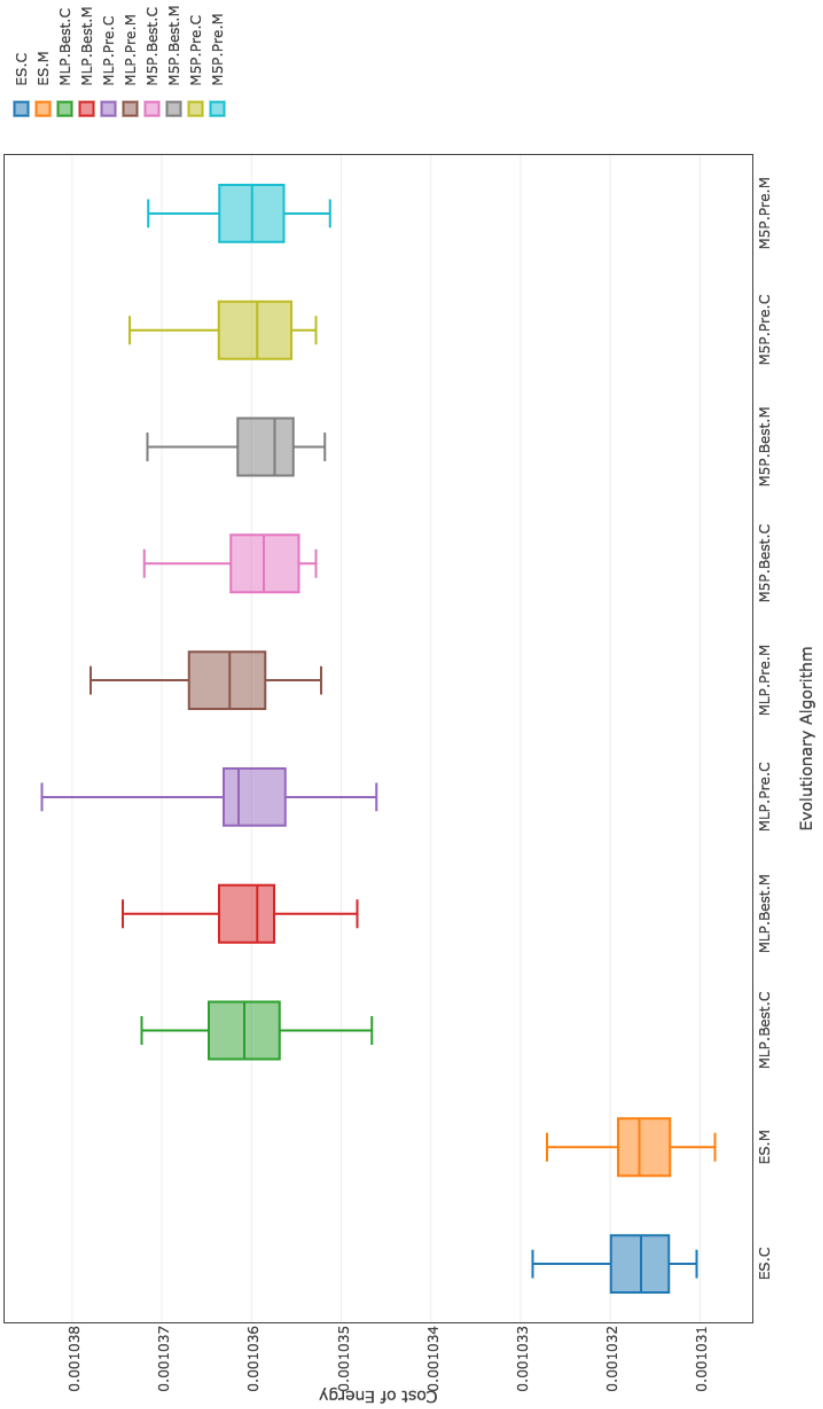

Figure 7.22: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario.

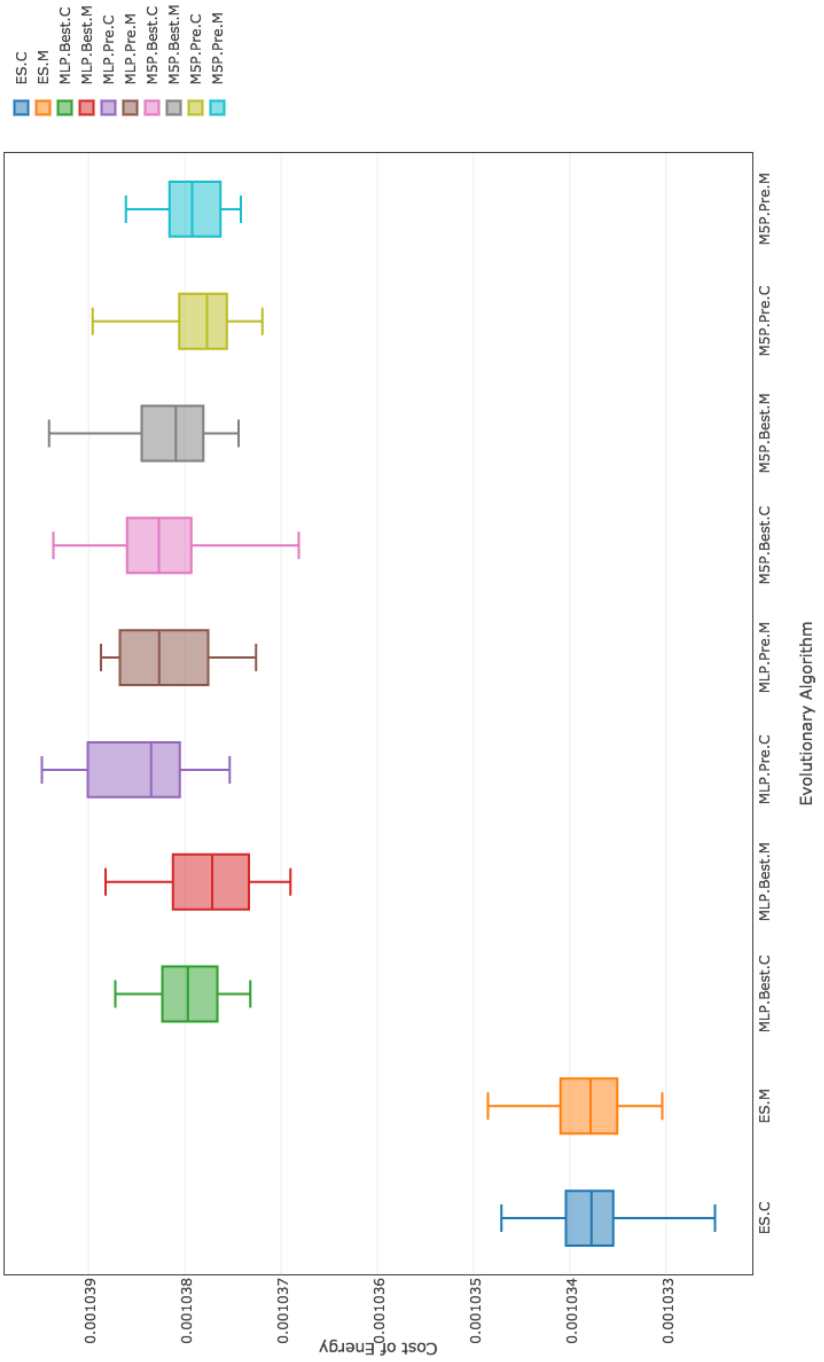

Figure 7.23: Cost of best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario.

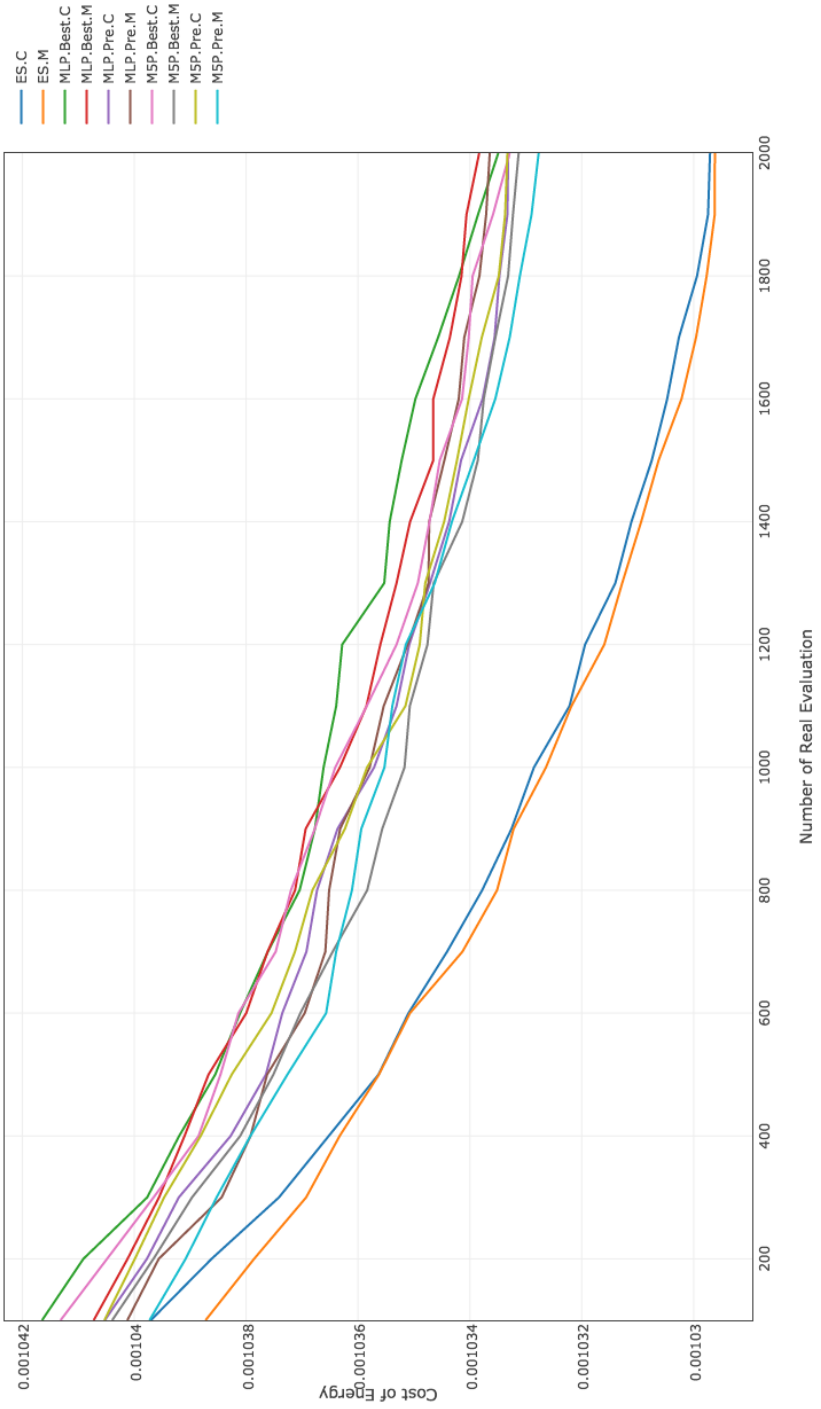

Figure 7.24: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(6, 12)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario.

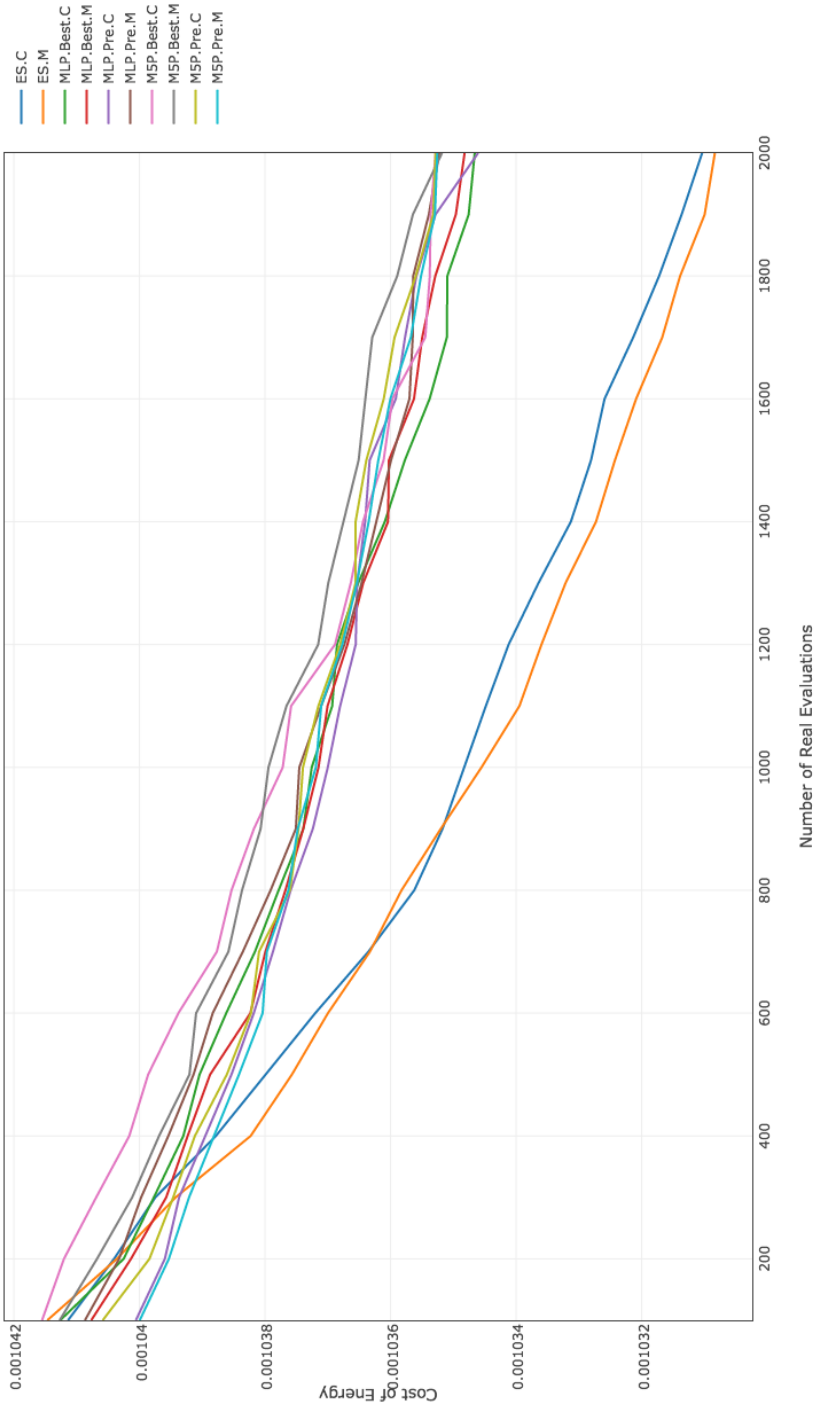

Figure 7.25: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(10, 20)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario.

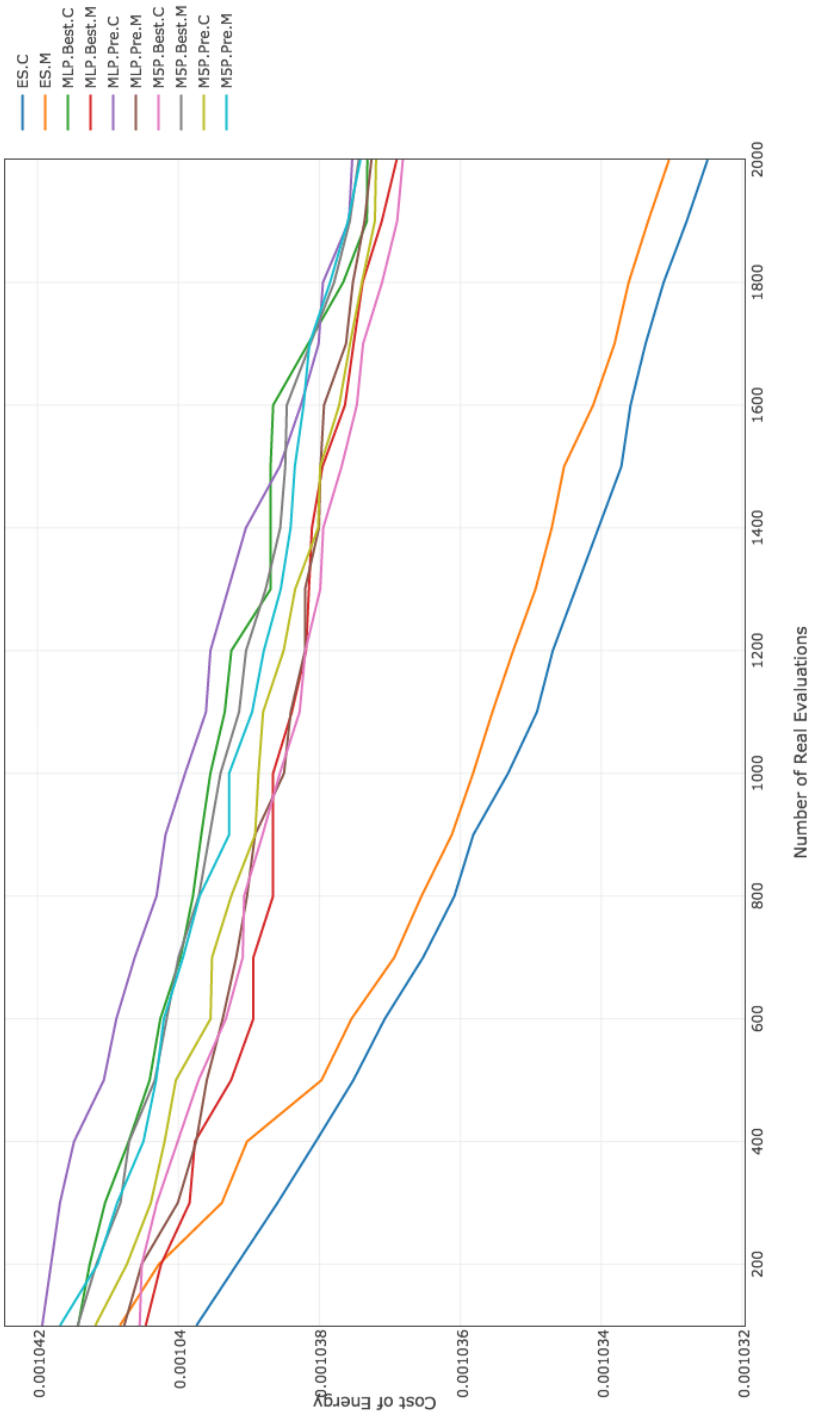

Figure 7.26: Convergence history of the best layouts found by different evolutionary algorithms (see Table 7.3) using $(20, 40)$ population configuration on the 2014 Wind Farm Layout Optimisation Comp 3 [105] wind farm scenario.

7.5 Summary of Evaluation Results

As can be observed, distributions are quite different for each algorithm. Unfortunately, the EA algorithm with no surrogate is clearly the winner on each scenario in terms of lowest overall cost. The EAs with no surrogate model significantly outperformed surrogate-assisted EAs on complex wind farm scenario Comp 1 [105] and Comp 3 [105] (440 and 1420 dimensions, respectively). On the simpler scenarios Kusiak & Song 1 [54] & 2 [54] (both are 200 dimensions), the difference of performance between EAs and surrogate-assisted EAs are smaller.

Out of 3,600 experimental runs on four scenarios, only the MLP-assisted $(6, 12)$-EA using Pre-selection strategy and BlockCopy Crossover operator obtained one layout with the lowest cost of energy compared to the rest of the algorithms on wind farm scenario Kusiak & Song 2 [54]. This phenomenon is not general, however, it demonstrates that the surrogate-assisted evolutionary algorithms may be promising for the WFLOP.

Overall the M5P surrogate model showed slightly better performance. The MLP-assisted EAs occasionally obtained better global solution compared to M5P assisted EAs. The possible reason for the poor performance of surrogate models is the limited number of real evaluations (2,000) and amount of training data (1,000 layouts with real fitness values). Poorly trained surrogate models cannot accurately approximate the real fitness function. Thus inaccurate surrogate models mislead the evolution algorithm to converge to a false optima. It might be helpful to increase the number of evaluation, but given the computational expense of the evaluation function and the fact that many other research works use similar criteria, we kept it to 2,000.

Speaking of the population configurations, the EAs with smaller population size yield better overall results. Using population-based approach can avoid local optima traps. However, the evaluations using larger population size are worse than smaller population size. The likely reason

for the poor performance of the larger population size is that increasing population size results in more exploration at the expense of exploitation.

The poor results of the surrogate-assisted algorithms are somewhat surprising, given the fact that the correlation coefficient can be as high as 0.9 according the initial experiments. The possible reason for that is the surrogates are doing interpolation during the initial experiments whereas the actual optimisation process employed the surrogates for extrapolation. The latter one is much harder since the surrogates are predicting the unknowns.

Tables 7.4 to 7.7 indicates that the surrogate-assisted algorithms consumed longer time given a fixed total number of real evaluations yet they produced poor final results. On the simpler wind farm scenarios, the population configuration has an significant impact on the non-surrogate evolutionary algorithms. When there are more wind turbines and obstacles are involved, the population configuration is less influential.

The initialisation of candidate wind farm layouts is important. The final results of the optimisation process may be influenced by the quality of the randomly generated initial wind farm layouts. For example, in Figures 7.8, 7.24 and 7.26, good fitness (low cost of energy) initial wind farm layouts are more likely to achieve better final results.

Chapter 8

Conclusions

To conclude, this thesis has investigated an evolutionary approach using surrogate models for the two-dimensional wind farm layout optimisation problem on four benchmark wind farm scenarios. Even the wake model and objective function we used are simplified, but they are still very computationally expensive. A wind farm layout dataset was created by using a biased random walk. The polar coordinates learning feature was selected based on a set of initial offline experiments. The MLP neural network and the M5P tree-based regression model were chosen as the surrogate model according to the initial experiment results. We used the BlockCopy operator in both mutation and crossover context. We added surrogate model re-training feature to the employed Pre-selection and Best surrogate management techniques.

Unfortunately, the overall performance of surrogate-assisted evolutionary strategies are not ideal compared to traditional evolutionary approaches in our experiments. Surrogate-assisted evolutionary approaches are still promising based on the possibility of more exploration of the search space. While the final wind farm layout solutions obtained using surrogate models are not better than the solutions obtained by using the EA without surrogate models, more research needs to be done.

In the literature, most of the surrogate-assisted evolutionary optimisation approaches are designed for relatively low-dimensional optimisa-

tion problems. However, the dimensionality of the four benchmark wind farm scenarios ranges from 200 to 1420 dimensions. The high input dimension of the objective function is the main reason why the constructed surrogate models may not be able to accurately approximate the real fitness function. Using inaccurate surrogate models may introduce false optimums, which mislead the evolutionary search. As a result, the high dimensionality of wind farm layouts caused the surrogate-assisted evolutionary algorithms having difficulties to converge.

Our current and ongoing work in this area is to develop higher accuracy surrogate models to improve the approximation of the real fitness function. We would like to improve the preliminary surrogate modelling process such as using a larger amount of training data, developing a more sophisticated structure of multilayer perceptron and other sophisticated surrogate models. These surrogates in the future may be used to help solve this important problem in renewable energy optimisation. Conversely, the WFLOP has been shown to an excellent high-dimensionality benchmark problem for testing surrogate-assisted evolutionary optimisation approaches.

Bibliography

[1] Alonso, J. J., LeGresley, P., Pereyra, V. (2009). Aircraft design optimization. *Mathematics and Computers in Simulation*, *79*(6), 1948–1958.

[2] Arabnia, M., Ghaly, W. (2009). A strategy for multi-point shape optimization of turbine stages in three-dimensional flow. In *ASME Turbo Expo 2009: Power for Land, Sea, and Air*, pp. 489–502. American Society of Mechanical Engineers.

[3] Atef, D., Osman, H., Ibrahim, M., Nassar, K. (2010). A simulation-based planning system for wind turbine construction. In *Proceedings of the Winter Simulation Conference*, WSC '10, pp. 3283–3294.

[4] Autodesk Education Community (2000). Orientation strategies for passive cooling. [Online; accessed 10-02-16]. `http://sustainabilityworkshop.autodesk.com/buildings/massing-orientation-cooling`

[5] AWS Truepower LLC. (2016). Openwind. [Online; accessed 17-02-16]. `http://software.awstruepower.com/openwind/`

[6] Azzouz, N., Bechikh, S., Ben Said, L. (2014). Steady state IBEA assisted by MLP neural networks for expensive multi-objective optimization problems. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 581–588. ACM.

[7] Baños, R., Manzano-Agugliaro, F., Montoya, F., Gil, C., Alcayde, A., Gómez, J. (2011). Optimization methods applied to renewable and sustainable energy: A review. *Renewable and Sustainable Energy Reviews*, *15*, 1753–1766.

[8] Banko, M., Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting on association for computational linguistics*, pp. 26–33. Association for Computational Linguistics.

[9] Bhattacharya, M. (2013). Evolutionary approaches to expensive optimisation. *International Journal of Advanced Research in Artificial Intelligence*, *2*(3), 53–59.

[10] Branke, J., Schmidt, C. (2005). Faster convergence by means of fitness estimation. *Soft Computing*, *9*(1), 13–20.

[11] Brower, M., Robinson, N. (2012). The openwind deep-array wake model: Development and validation. *AWS Truepower*.

[12] Bull, L. (1999). On model-based evolutionary computation. *Soft Computing*, *3*(2), 76–82.

[13] Burton, T., Sharpe, D., Jenkins, N., Bossanyi, E. (2001). *Wind energy handbook*. John Wiley & Sons.

[14] Chen, L., MacDonald, E. (2012). Considering landowner participation in wind farm layout optimization. *Journal of Mechanical Design*, *134*(8), 084506–084506.
`http://dx.doi.org/10.1115/1.4006999`

[15] Chowdhury, M., Alouani, A., Hossain, F. (2010). Comparison of ordinary kriging and artificial neural network for spatial mapping of arsenic contamination of groundwater. *Stochastic Environmental Research and Risk Assessment*, *24*(1), 1–7.

[16] D'Angelo, S., Minisci, E. A. (2005). Multi-objective evolutionary optimization of subsonic airfoils by kriging approximation and evolu-

tion control. In *2005 IEEE congress on evolutionary computation*, vol. 2, pp. 1262–1267. IEEE.

[17] Dasari, S. K., Lavesson, N., Andersson, P., Persson, M. (2015). Tree-based response surface analysis. In *International Workshop on Machine Learning, Optimization and Big Data*, pp. 118–129. Springer.

[18] DNV GL Group (2016). Wind Atlas Analysis and Application Program (WAsP). [Online; accessed 16-02-16]. `http://www.wasp.dk`

[19] Douguet, D. (2010). e-LEA3D: a computational-aided drug design web server. *Nucleic acids research*, p. gkq322.

[20] DTU Wind Energy (2016). WindPRO. [Online; accessed 16-02-16]. `http://www.emd.dk/windpro/`

[21] Eigen, M. (1973). *Ingo Rechenberg Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologishen Evolution*. mit einem Nachwort von Manfred Eigen, Friedrich Frommann Verlag, Struttgart-Bad Cannstatt.

[22] Ekonomou, L., Lazarou, S., Chatzarakis, G., Vita, V. (2012). Estimation of wind turbines optimal number and produced power in a wind farm using an artificial neural network model. *Simulation Modelling Practice and Theory*, *21*(1), 21 – 25.

[23] Emami, A., Noghreh, P. (2010). New approach on optimization in placement of wind turbines within wind farm by genetic algorithms. *Renewable Energy*, *35*(7), 1559–1564.

[24] EMD International A/S (2016). EMD International A/Swindfarmer. [Online; accessed 16-02-16]. `https://www.dnvgl.com/services/windfarmer-3766`

[25] Frandsen, S. (1992). On the wind speed reduction in the center of large clusters of wind turbines. *Journal of Wind Engineering and Industrial Aerodynamics*, *39*(1-3), 251–265.

[26] Garza, J., Blatt, A., Gandoin, R., Hui, S. (2011). Evaluation of two novel wake models in offshore wind farms. In *Proceedings of the European Wind Energy Associate Offshore Conference*.

[27] Global Wind Energy Council (2015). *Global Wind Energy Outlook 2015*.

[28] González, J. S., Payán, M. B., Santos, J. M. R., González-Longatt, F. (2014). A review and recent developments in the optimal wind-turbine micro-siting problem. *Renewable and Sustainable Energy Reviews*, *30*, 133–144.

[29] González, J. S., Rodríguez, A. G., Mora, J. C., Payán, M. B., Santos, J. R. (2011). Overall design optimization of wind farms. *Renewable Energy*, *36*(7), 1973–1982.

[30] González, J. S., Rodriguez, A. G. G., Mora, J. C., Santos, J. R., Payan, M. B. (2010). Optimization of wind farm turbines layout using an evolutive algorithm. *Renewable Energy*, *35*(8), 1671–1681.

[31] Grady, S., Hussaini, M., Abdullah, M. M. (2005). Placement of wind turbines using genetic algorithms. *Renewable energy*, *30*(2), 259–270.

[32] Grefenstette, J. J., Fitzpatrick, J. M. (1985). Genetic search with approximate function evaluation. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 112–120. Hillsdale, NJ, USA: L. Erlbaum Associates Inc. ISBN 0-8058-0426-9. `http://dl.acm.org/citation.cfm?id=645511.657078`

[33] Gu, H., Wang, J., Lin, Q., Gong, Q. (2015). Automatic contour-based road network design for optimized wind farm micrositing. *IEEE Transactions on Sustainable Energy*, *6*(1), 281–289.

[34] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, *11*(1).

[35] Han, Z.-H., Zhang, K.-S. (2012). *Surrogate-based optimization*. INTECH Open Access Publisher.

[36] Heller, A. (2010). Wind turbine. [Online; accessed 22-11-15]. `https://str.llnl.gov/AprMay10/mirocha.html`

[37] Herbert-Acero, J. F., Probst, O., Réthoré, P.-E., Larsen, G. C., Castillo-Villar, K. K. (2014). A review of methodological approaches for the design and optimization of wind farms. *Energies*, *7*(11), 6930. doi:10.3390/en7116930. `http://www.mdpi.com/1996-1073/7/11/6930`

[38] Hinton, G. E. (1989). Connectionist learning procedures. *Artificial intelligence*, *40*(1), 185–234.

[39] Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press.

[40] Huang, G.-B. (2003). Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, *14*(2), 274–281.

[41] Huang, H.-S. (2007). Distributed genetic algorithm for optimization of wind farm annual profits. In *Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on*, pp. 1–6. IEEE.

[42] Hüsken, M., Jin, Y., Sendhoff, B. (2005). Structure optimization of neural networks for evolutionary design optimization. *Soft Computing*, *9*(1), 21–28.

[43] J. Forrester, A. I., Keane, A. J., Bressloff, N. W. (2006). Design and analysis of "Noisy" computer experiments. *AIAA journal*, *44*(10), 2331–2339.

[44] Jensen, N. O. (1983). *A note on wind generator interaction*.

[45] Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, *9*(1), 3–12.

[46] Jin, Y. (2011). Surrogate-assisted evolutionary computation: recent advances and future challenges. *Swarm and Evolutionary Computation*, *1*, 61–70.

[47] Jin, Y., Olhofer, M., Sendhoff, B. (2000). On evolutionary optimization with approximate fitness functions. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, pp. 786–793. Morgan Kaufmann Publishers Inc.

[48] Jin, Y., Olhofer, M., Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. *Evolutionary Computation, IEEE Transactions on*, *6*(5), 481–494.

[49] Jin, Y., Sendhoff, B. (2009). A systems approach to evolutionary multiobjective structural optimization and beyond. *IEEE Computational Intelligence Magazine*, *4*(3), 62–76.

[50] Katic, I. (1993). Program park, calculation of wind turbine park performance, release 1.3++. *Risø National Laboratory, Rosklide*.

[51] Katic, I., Højstrup, J., Jensen, N. O. (1986). A simple model for cluster efficiency. In *European Wind Energy Association Conference and Exhibition*, pp. 407–410.

[52] Kbiob, D. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of Chemical, Metallurgical, and Mining Society of South Africa*.

[53] Ko, H.-S., Jatskevich, J. (2007). Power quality control of wind-hybrid power generation system using fuzzy-LQR controller. *IEEE Transactions on energy conversion*, *22*(2), 516–527.

[54] Kusiak, A., Song, Z. (2010). Design of wind farm layout for maximum wind energy capture. *Renewable Energy*, *35*, 685–694.

[55] Kusiak, A., Zheng, H. (2010). Optimization of wind turbine energy and power factor with an evolutionary computation algorithm. *Energy*, *35*(3), 1324–1332.

[56] Kusiak, A., Zheng, H., Song, Z. (2010). Power optimization of wind turbines with data mining and evolutionary computation. *Renewable Energy*, *35*(3), 695–702.

[57] Kwong, W., Zhang, P. Y., Romero, D. (2012). Wind farm layout optimization considering energy generation and noise propagation. In *Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2012*.

[58] Lackner, M. A., Elkinton, C. N. (2007). An analytical framework for offshore wind farm layout optimization. *Wind Engineering*, *31*(1), 17–31.

[59] Langdon, W. B., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., et al. (2002). *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers.

[60] Laumanns, M., Zitzler, E., Thiele, L. (2000). A unified model for multi-objective evolutionary algorithms with elitism. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, pp. 46–53. IEEE.

[61] Lian, Y., Liou, M.-S. (2005). Multi-objective optimization of transonic compressor blade using evolutionary algorithm. *Journal of Propulsion and Power*, *21*(6), 979–987.

[62] Lian, Y., Liou, M.-S. (2005). Multiobjective optimization using coupled response surface model and evolutionary algorithm. *AIAA journal*, *43*(6), 1316–1325.

[63] Lim, D., Ong, Y.-S., Jin, Y., Sendhoff, B. (2006). Trusted evolutionary algorithm. In *2006 IEEE International Conference on Evolutionary Computation*, pp. 149–156. IEEE.

[64] Liu, B., Zhang, Q., Gielen, G. G. (2014). A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. *IEEE Transactions on Evolutionary Computation*, *18*(2), 180–192.

[65] Loshchilov, I., Schoenauer, M., Sebag, M. (2010). A mono surrogate for multiobjective optimization. In *Proceedings of the 12th annual conference on Genetic and Evolutionary Computation*, pp. 471–478. ACM.

[66] Lückehe, D., Wagner, M., Kramer, O. (2015). On evolutionary approaches to wind turbine placement with geo-constraints. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15.

[67] Luke, S. (2014). *Essentials of Metaheuristics*. Online version 2.1 edn.

[68] Mallipeddi, R., Lee, M. (2015). An evolving surrogate model-based differential evolution algorithm. *Applied Soft Computing*, *34*, 770 – 787. doi:http://dx.doi.org/10.1016/j.asoc.2015.06.010. `http://www.sciencedirect.com/science/article/pii/S1568494615003592`

[69] Manwell, J. F., McGowan, J. G., Rogers, A. L. (2010). *Wind energy explained: theory, design and application*. John Wiley & Sons.

[70] Marmidis, G., Lazarou, S., Pyrgioti, E. (2008). Optimal placement of wind turbines in a wind park using monte carlo simulation. *Renewable energy*, *33*(7), 1455–1460.

[71] Martínez, S. Z., Coello, C. A. C. (2010). A memetic algorithm with non gradient-based local search assisted by a meta-model. In *International Conference on Parallel Problem Solving from Nature*, pp. 576–585. Springer.

[72] Martinez-Cesena, E. A., Mutale, J. (2012). Wind power projects planning considering real options for the wind resource assessment. *IEEE Transactions on Sustainable Energy*, *3*(1), 158–166.

[73] Mayo, M., Zheng, C. (2016). Blockcopy-based operators for evolving efficient wind farm layouts. In *IEEE Congress on Evolutionary Computation, CEC '16.*

[74] Mora, J. C., Barón, J. M. C., Santos, J. M. R., Payán, M. B. (2007). An evolutive algorithm for wind farm optimal design. *Neurocomputing, 70*(16), 2651–2658.

[75] Mosetti, G., Poloni, C., Diviacco, B. (1994). Optimization of wind turbine positioning in large wind farms by means of a genetic algorithm. *Journal of Wind Engineering and Industrial Aerodynamics, 51*(1), 105–116.

[76] Muljadi, E., McKenna, H. E. (2001). Power quality issues in a hybrid power system. In *Industry Applications Conference, 2001. Thirty-Sixth IAS Annual Meeting. Conference Record of the 2001 IEEE*, vol. 2, pp. 773–781. IEEE.

[77] Mustakerov, I., Borissova, D. (2010). Wind turbines type and number choice using combinatorial optimization. *Renewable Energy, 35*(9), 1887–1894.

[78] Neubert, A., Shah, A., Schlez, W. (2010). Maximum yield from symmetrical wind farm layouts. In *10th German Wind Energy Conference (DEWEK).*

[79] Ong, Y. S., Nair, P. B., Keane, A. J. (2003). Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal, 41*(4), 687–696.

[80] Ozturk, U. A., Norman, B. A. (2004). Heuristic methods for wind energy conversion system positioning. *Electric Power Systems Research, 70*(3), 179–185.

[81] Pagano, A., Federico, L., Barbarino, M., Guida, F., Aversano, M. (2008). Multi-objective aeroacoustic optimization of an aircraft propeller. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia Canada.*

[82] Patvardhan, C., Bansal, S., Srivastav, A. (2015). Quantum-inspired evolutionary algorithm for difficult knapsack problems. *Memetic Computing*, *7*(2), 135–155.

[83] Pierret, S., Van den Braembussche, R. (1998). Turbomachinery blade design using a navier-stokes solver and artificial neural network. In *ASME 1998 International Gas Turbine and Aeroengine Congress and Exhibition*, pp. V001T01A002–V001T01A002. American Society of Mechanical Engineers.

[84] Ponsich, A., Jaimes, A. L., Coello, C. A. C. (2013). A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications. *IEEE Transactions on Evolutionary Computation*, *17*(3), 321–344.

[85] Quinlan, R. J. (1992). Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pp. 343–348. Singapore: World Scientific.

[86] Rai, M. M. (2005). Hybrid neural network and support vector machine method for optimization. US Patent 6,961,719.

[87] Ratle, A. (1998). Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *International Conference on Parallel Problem Solving from Nature*, pp. 87–96. Springer.

[88] Rivas, R. A., Clausen, J., Hansen, K. S., Jensen, L. E. (2009). Solving the turbine positioning problem for large offshore wind farms by simulated annealing. *Wind Engineering*, *33*(3), 287–297.

[89] Samorani, M. (2013). The wind farm layout optimization problem. In Pardolas, P. (Ed.), *Handbook of Wind Power Systems*, pp. 21–38. Springer-Verlag.

[90] Serrano-González, J., Burgos-Payán, M., Riquelme-Santos, J. (2013). Design of neighboring large offshore wind farms: A game

theory approach. In *Proceedings of the European Wind Energy Conference and Exhibition*.

[91] Siemens Co. (2000). The middelgrunden offshore wind farm. [Online; accessed 10-01-16].
`http://www.renewable-technology.com/projects/middelgrunden-wind-farm-denmark/middelgrunden-wind-farm-denmark1.html`

[92] Siemens Co. (2007). Wind energy industry-standard software. [Online; accessed 10-01-16].
`https://web.archive.org/web/20070706184255/http://www.cece.dk/EE0911AA-D9A1-49E8-9CA2-332E37BBA568`

[93] Şişbot, S., Turgut, Ö., Tunç, M., Çamdalı, Ü. (2010). Optimal positioning of wind turbines on Gökçeada using multi-objective genetic algorithm. *Wind Energy*, *13*(4), 297–306.

[94] Song, W., Keane, A. J. (2007). Surrogate-based aerodynamic shape optimization of a civil aircraft engine nacelle. *AIAA journal*, *45*(10), 2565–2574.

[95] Stevens, M., Smulders, P. (1979). The estimation of the parameters of the weibull wind speed distribution for wind energy utilization purposes. *Wind engineering*, *3*, 132–145.

[96] Sun, C., Ding, J., Zeng, J., Jin, Y. (2016). A fitness approximation assisted competitive swarm optimizer for large scale expensive optimization problems. *Memetic Computing*, pp. 1–12.

[97] Vermeer, L., Sørensen, J. N., Crespo, A. (2003). Wind turbine wake aerodynamics. *Progress in aerospace sciences*, *39*(6), 467–510.

[98] Vestas Wind Systems A/S (2016). Wind turbine v63/1500 (vestas).

[99] Wagner, M., Day, J., Neumann, F. (2013). A fast and effective local search algorithm for optimizing the placement of wind turbines. *Renewable Energy*, *51*, 64–70.

[100] Wang, L., Tan, A. C., Gu, Y., Yuan, J. (2015). A new constraint handling method for wind farm layout optimization with lands owned by different owners. *Renewable Energy, 83*, 151–161.

[101] Wang, Y., Witten, I. H. (1997). Induction of model trees for predicting continuous classes. In *Poster papers of the 9th European Conference on Machine Learning*. Springer.

[102] Weibull, W. (1951). A statistical distribution function of wide applicability, presented to the american society of mechanical engineers. *Atlantic City, NJ, 23*, 981–997.

[103] Wilcox, D. C., et al. (1998). *Turbulence modeling for CFD*, vol. 2. DCW industries La Canada, CA.

[104] Williams, C. K., Rasmussen, C. E. (1996). Gaussian processes for regression.

[105] Wilson, D. (2015). Wind Farm Layout Optimization Competition. [Online; accessed 31-10-15]. `https://www.irit.fr/wind-competition/`

[106] WindSim (2016). Windsim. [Online; accessed 16-02-16]. `https://www.windsim.com`

[107] Zhang, K.-s., Han, Z.-h., Li, W.-j., Song, W.-p. (2008). Coupled aerodynamic/structural optimization of a subsonic transport wing using a surrogate model. *Journal of Aircraft, 45*(6), 2167–2171.

[108] Zhou, Z., Ong, Y. S., Nair, P. B., Keane, A. J., Lum, K. Y. (2007). Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 37*(1), 66–76.

[109] Zitzler, E., Deb, K., Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation, 8*(2), 173–195.