Wind Farm Layout Optimization using Distributed Genetic Algorithms

Second title for thesis

Helene Tangen

NTNU – Trondheim
Norwegian University of
Science and Technology

Department of Computer and Information Science
Faculty of Information Technology, Mathematics and
Electrical Engineering

Trondheim, June 2015

Contents

Chapter 1

Introduction

1.1 Introduction

1.1.1 Motivation and Background

Transitioning from non-renewable energy sources to renewable energy sources is one of the largest, if not the larges political challenge of today. Renewable energy is less polluting that non-renewable energy and should therefore be preferred. However, renewable energy sources make up only 17.05 percent of the worlds energy sources as of 26th of May 2015 (renewableenergyworld.com).

Wind turbine technology is a promising source of renewable energy. Wind turbine technology advances has led to wind turbines able to produce more energy to lower costs. However, wind turbines still produces less energy than predicted because of the wake effect [Samorani, 2013]. For wind energy to become a bigger player in the worlds energy sources, sophisticated methods for wind turbine placement in wind farms needs to be developed so that each turbine produces as much energy as possible.

Wind turbine positioning is hard to optimize analytically. Fortunately, use of genetic algorithms shows promising results. As more advanced approaches to evaluate layouts are developed, and more realistic constrains are introduced, more sophisticated genetic algorithms are required.

To come up with more sophisticated genetic algorithms for solving the wind farm layout optimization problem, the annual Genetic and Evolutionary Computation Conference (GECCO), launched a competition where different contestants provide their own implementation of a genetic algorithm. The goal of the competition is to bring more realistic problems to algorithm developers, and to create an open source library useful beyond the scope of the competition (http://www.irit.fr/wind-competition/).

Wind parameters and evaluation mechanism are provided by GECCO, therefore the goal of this project will be to optimize the genetic algorithm for solving the problem not wind farm parameters and models. However, knowledge of wind turbines, wind farm layout and wake models are useful in understanding the project and is therefore introduced in the background section.

1.1.2 Goal and Research Questions

This section states the goal statement and research questions that will be investigated in this thesis.

Goal statement

> *The project goal is to investigate the advantages of using distributed genetic algorithms to optimizing wind farm layout, i.e. solving the wind farm layout optimization problem.* [Samorani, 2013]

The performance of distributed genetic algorithms will be studied and compared to the performance of a simple genetic algorithm (not distributed) as well as to each other, with the goal of answering the research questions stated below.

Research question 1

> *Can distributed genetic algorithms improve the quality of the solution to the wind farm layout optimization problem as compared to simple genetic algorithm.*

Research question 2

Which distributed genetic algorithm works best for the wind farm layout optimization problem? What properties are essential for its success?

Chapter 2

Background

2.1 The wind farm layout optimization problem

The goal of this section is to give the reader an understanding of the wind farm layout optimization problem, and explain the key factors that makes the problem so complex.

2.1.1 Definition of the wind farm layout optimization problem

An overview of the wind farm layout optimization problem is presented by Samorani [Samorani, 2013]. Grouping of wind turbines in a wind farm decreases installation and maintenance cost. However, positioning of wind turbines in a farm also introduces new challenges. The power produced by wind turbines is largely dependent on wind speed, therefore it is important that the wind speed that hits a wind turbine is as large as possible. The main challenge for wind farms is that a wind turbine positioned in front of another wind turbine will cause a wake of turbulence, meaning that the wind speed that hits the second wind turbine will be decreased. This effect is called "wake effect", and will be explained later. Since the goal is to produce as much power as possible it is very important to position the wind turbines so that the wake effect is minimal. Samorani states the wind farm layout optimization problem like this "The wind farm layout optimization problem consists of finding the turbine positioning (wind farm layout) that maximizes the expected power production". However, in this thesis, the problem formulation will be extended to include cost constraints and also the problem of deciding the number of wind turbines, not just their positions. A formal definition is given below

> *"The wind farm layout optimization problem consists of finding the number of turbines and turbine positioning (wind farm) that maximizes the expected power production within a given budget."*

2.1.2 Challenges of wind farm construction

Samorani gives an overview of the main challenges of wind farm construction. First, a suitable site has to be found, meaning a site with good wind conditions. Sites are classified in 7 different wind power classes, where sites with power class 4 or higher are suitable for hosting a wind farm with today's turbine technology. But, even though the wind farm has the required wind conditions, it might not be suitable for hosting a find farm after all, because it might be far from the electronic grid, so that connecting it to it would be to costly, or it could require costly road work because current roads cannot handle the transportation trucks.

Second, land owner has to be contacted and convinced that hosting a wind farm on their land is a good idea. Land owners usually gets a percentage of the wind farm profit. This phase of contract negotiation usually takes a few months. At the same time, wind distribution needs to be measured as accurately as possible. This step is extremely important, since the layout of the farm is optimized based on the measured wind distribution. Getting enough data to capture the wind distribution can take a few months if wind conditions are similar all year long, but if the wind conditions vary extensively over the year this step can take a few years.

An evenly important step is to decide on which turbines to buy for the wind farm. Larger turbines usually generate more power, but they are also more expensive than smaller ones. There is therefore a trade off between the cost and power production. Realistic estimation of maintenance cost is also crucial in deciding on turbine type. In [Samorani, 2013] the number of wind turbines are also decided in this step, but in this project, deciding the number of turbines is included in the wind farm layout optimization problem and will therefore be part of the next step.

After the site is found, turbine type is decided and wind distribution is measured, the layout optimization can begin. Layout optimization faces different challenges, such as positions of the terrain that contain obstacles so that turbines cannot be positioned there. There are also constraint on how close turbines can be positioned, according to [Şişbot et al., 2010], the minimum spacing rule states that the minimum distance between turbines is 8D in prevailing wind direction, and 2D in cross wind direction, where D is the rotor diameter. However, the greatest challenge of wind farm layout optimization is the wake effect. As mentioned above, the wake effect is the effect of reduced wind speed in the wake behind a wind turbine. Samorani explains the wake effect using the Jensen wake model [Jensen, 1983], other wake models exist, but most research in wind farm layout optimization use the Jensen model because it is quite accurate and simple. The Jensen model will also be used briefly in this project, to give an intuitive explanation of the wake effect.

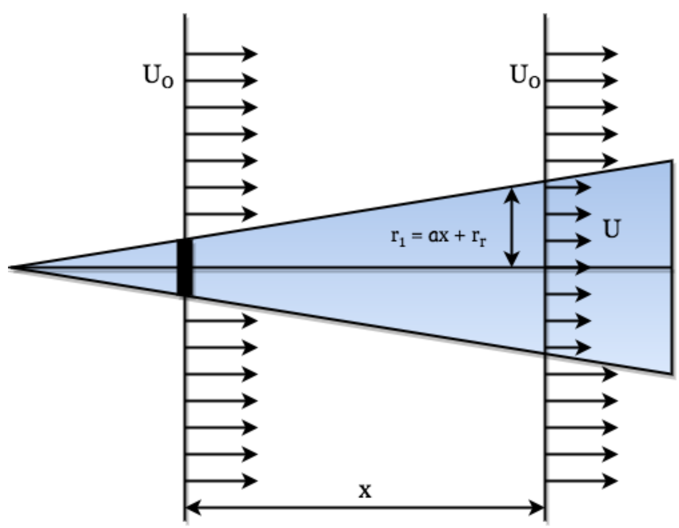


Figure 2.1: The wake effect [Samorani, 2013]

In figure 2.1 the small black rectangle represents a wind turbine, and the blue area behind it illustrates the area that is affected by the turbulence created by the wind turbine. In the figure, the wind is blowing from left to right with uniform wind speed of U_0. As the wind hits the wind turbine it creates a wake of turbulence behind it so that the wind speed at distance x behind the wind turbine is $U < U_0$. The area behind the wind turbine that is affected by the wake at distance x has the radius $r_1 = \alpha x + r_r$ where r_r is the rotor radius and α is the entrainment constant constant which decides how fast the wake expands. For a detailed, mathematical explanation of the Jensen model and other wake models see [Jensen, 1983], [Liang et al., 2014].

In summary, construction of a wind farm is a complicated, time consuming process. In order to even start the layout optimization consecutive important decisions has to be made. The layout optimization is dependent on turbine cost, terrain parameters, wind conditions and turbine positioning. Finding the optimal layout is a non-linear, complex problem that only sophisticated algorithms can solve.

2.2 Genetic Algorithms

This section first explains the simple genetic algorithm (SGA), invented by Holland [1992], to give an understanding of how the genetic algorithm works. If the reader is familiar with the genetic algorithm he or she can skip the first part. Next, the five different distributed genetic algorithms that will be implemented in this thesis are explained, these are all taken from Gong et al. [2015]. If not otherwise stated, the first part (SGAs) is based on [Holland, 1992] and [Goldberg, 2005], and the second part (DGAs) is based on [Gong et al., 2015].

2.2.1 Simple Genetic Algorithms (SGAs)

Genetic algorithms are probabilistic search algorithms inspired by evolution. Figure 2.2 gives an intuitive explanation of how the algorithm works. The genetic algorithm operates on a population of individuals each representing a solutions to a problem. Usually, the initial population consist of randomly generated individuals which becomes the first child population. For each generation, the child population are evaluated based on some predefined fitness function (objective function), and the fittest individuals are selected as parents for the next generation. Next, the parent population produces a new child population based on different reproduction schemes, such as recombination of parent genes to form child genes. Some genes are also mutated in the process. Finally, the next generation of child solutions are generated and the process starts again.

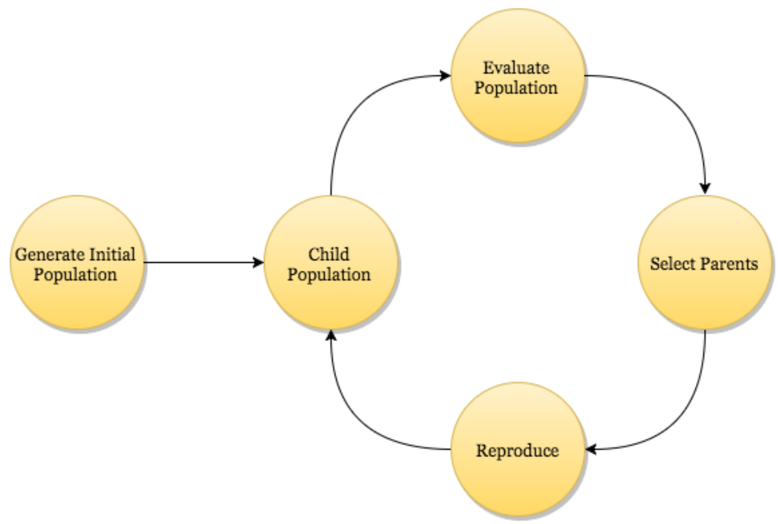


Figure 2.2: Overview of the phases of the genetic algorithm.

Inspired by survival of the fittest, the population evolves into a population of better solutions to the given problem. Two key properties are crucial for the utilization and success of the genetic algorithm; (1) there has to be a way to evaluate the fitness of the solutions, and (2) there needs to be a way to represent individuals so that genetic operations can be performed. Examples of representation, fitness calculation, selection processes, and genetic operations will be given below. Note that there exist numerous different selection schemes and ways to perform mutation and crossover (genetic operations), but here, only the types that are used in the given thesis are presented.

2.2.1.1 Representation

In genetics, they call an organisms hereditary information for its genotype, and its observable properties its phenotype. For example, the hereditary information in your genes (genotype) are responsible for your eye color (phenotype). The genetic search algorithm usually works on genotypes represented as bit strings. Goldberg [2005] explained this with a simple example. Let's say the objective function that we want to find an optimal solution for is x^2 for $x \in \{0, 31\}$. Then we can generate genotypes for the random solutions using bit strings of size 5, each representing a phenotype value between 0 and 31. Figure 2.3 displays the genotype and phenotype for four randomly generated individuals. Here, the phenotypes are just the genotypes on decimal form, but in other problems the phenotype could be everything from eye color to a wind farm.

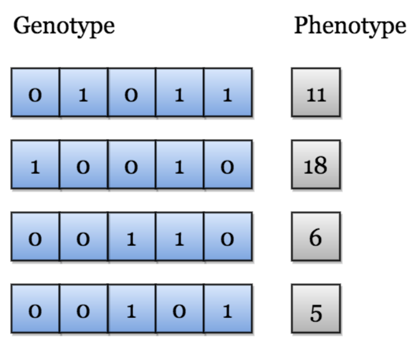


Figure 2.3: Genotypes and phenotypes for four individuals where the phenotype is the decimal value of the genotype.

2.2.1.2 Selection

Selection is the process of selecting which individuals from a given population that will be the parents of the next generation. The simplest form of selection is called *elitist selection*, meaning the n individuals from the populations are selected. Unfortunately, this selection strategy often leads to premature convergence of non optimal solutions. It is important to prioritize exploration, at least in the beginning of the search , otherwise, parts of the search space that could have lead to the optimal solution is cut off too soon. To cope with this problem *controlled elitist selection* schemes are preferred. A very popular selection strategy is *tournament selection* [Razali and Geraghty, 2011]. In tournament selection, groups of n individuals are randomly drawn from the population and the best (fittest) individual from the group is chosen as the tournament winner, and is therefore selected. Figure 2.4 illustrates how tournament selection works. In the example, n is equal to 3, therefore the three individuals with fitness 9, 4 and 6 are randomly drawn from the population. The individual with fitness 9 wins the tournament and is chosen for reproduction.

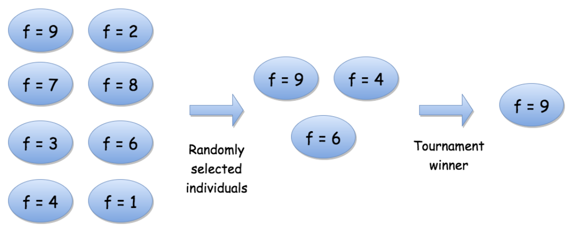


Figure 2.4: Tournament selection. A group of three individuals are randomly drawn from the pool of all individuals. The best individual in the group, the one with fitness 9, is selected for reproduction [Razali and Geraghty, 2011].

By varying the value of n you can control how much exploration your algorithm should do. If n is equal to the population size, this is elitist selection, if n is equal to 1 however the search is completely random. This means that low values of n leads to more exploration of the search space, and higher values of n leads to faster convergence. These properties makes it desirable to vary the value of n during the genetic search so that exploration is prioritized at the beginning of the search, while exploitation is prioritized at the end.

2.2.1.3 Crossover

Crossover means combining genes of a parent solution to produce a child. The mutation scheme used in this thesis uses a crossover method called uniform crossover. For each gene of the child solution there is a 50% chance the gene will be copied from the first parent and a 50% chance that the gene will be copied from the second parent. Figure 2.5 shows how uniform crossover works. As can be seen, the first gene is taken from parent one, the second gene from parent two, the third and forth gene from parent one and so on.

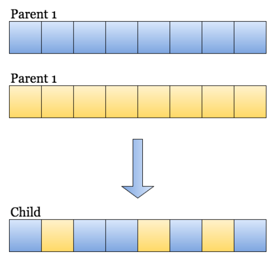


Figure 2.5: Uniform crossover. A child genotype is created by a combination of the genotypes of both parents. Each gene is drawn from one of the parents with equal probability.

2.2.1.4 Mutation

In biology, mutation is defined as permanent alteration in the DNA sequence that makes up a gene. When the genetic algorithm works on genotypes of bit strings the process consists of simply flipping bits. Mutation is usually implemented by having a given probability of each value in the genotype being flipped as shown in figure 2.6.

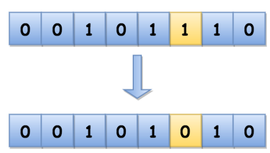


Figure 2.6: Mutation of a single bit. The bit in position 6 at the upper bit string has the value 1 before the mutation, while after mutation the value if flipped into 0.

Mutation is important because without mutation a population can converge to a population of individuals where each genotype has the same value at a given position. Since every individual has the same value in their genotype, reproduction will never be able to make a new individual that doesn't also have the same value at the same position. With mutation however, there is always a probability of the value being flipped, mutation is therefore crucial for maintaining diversity in the population.

Even though mutation is important, the probability of mutation needs to be kept low. If the mutation rate is very high, the genotype of a new individual will almost be a random bit string. Remember that a new individual is made by reproduction between two individuals with high fitness in the previous population, if mutation heavily changes the new individual, it will not inherit the good features of its parents and the whole point of evolutionary search will be gone.

2.2.2 Distributed Genetic Algorithms (DGAs)

On of the main challenges of simple genetic algorithms is keeping diversity in the population long enough so that the population do not converge to a sub-optimal solution. By distributing the population, the population is able to explore different solution paths, even some that does not look that good at first, and consequently get the opportunity to find better, more sophisticated solutions.

The goal of this thesis is to investigate and compare the performance of different distributed genetic algorithms in solving the wind farm layout optimization problem. In order to give an answer to the research questions, different distributed genetic algorithms needs to be implemented and tested. This section introduces five different population-distributed algorithms presented by Gong et al. [2015]. These will all be implemented and tested in this thesis.

2.2.2.1 The Master-Slave Model

The master-slave model is not really a distributed genetic algorithm, but a simple genetic algorithm where the main operations of the algorithm are distributed between different processors. It will be implemented in this thesis for two reasons; (1) distributing tasks between different processors gives a faster-running algorithm than an simple genetic algorithm, (2) results obtained will be the same as results obtained by a simple genetic algorithm, and can therefore be used to compare against true distributed algorithms. The master-slave model is displayed in figure 2.7.

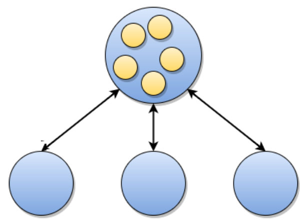


Figure 2.7: Master-slave model. The master process distributes the population to different slave processes, which calculate the fitness of each individual and return the results to the master process [Gong et al., 2015].

When the master-slave model is used, the main loop is taken care of by the master process, however the most expensive operation in the genetic algorithm, calculation of fitness, is distributed to different slave processes. Each slave simply calculate the fitness of the individuals received from the master, and return the calculated fitness to the master.

2.2.2.2 The Island Model

In the island model, the population is divided into sub populations that are distributed onto different islands. By letting each population evolve separately different islands can explore different solutions. Figure 2.8 displays a population divided into four sub populations.

According to Huang [2007], six parameters needs to be specified when using the island model. First of all, the deme size needs to be specified; the number of individuals on each island (deme). Second, one needs to decide on the number of demes. In figure 2.8 the deme size is five and four demes are used. Third, the topology must be specified; the allowed routes to migrate from one population to another. Numerous topologies can be used. In figure 2.8 the arrows are the legal migration routes, since the topology forms a circle it is called a ring topology. The forth and fifth parameters listed by Huang are migration rate and migration interval, meaning the number of individuals that migrate from one population to another and the number of generations between each migration respectively. These parameters are very

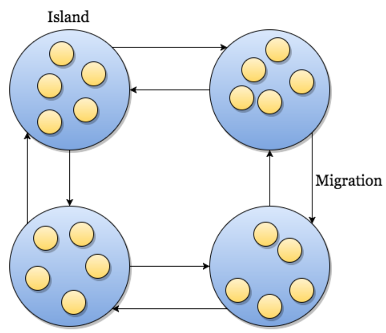


Figure 2.8: An island model using a ring topology with four demes of size five. [Gong et al., 2015]

important since they largely affect the time the population gets to explore different solutions before the best solutions from some of the demes takes over the population. Sixth, the policy of selection emigrants and how to replace existing individuals with new migrants.

The above parameters must be given careful thought when implementing the island model, but as Gong explains, they are not the only ones. If the island model is implement in parallel one also have to decide if the migration is synchronous or asynchronous. Synchronous migration means that all migration is performed at the same time; at a specific generation, while as asynchronous migration is used migration can be performed whenever one of the parallel processes is ready. Additionally, it has to be decided if the island model is homogeneous or heterogeneous. By a homogeneous island model, Gong means an island model where each sub population use the same selection strategy, genetic operations and fitness function, while as an heterogeneous island model can implement different settings for different sub populations.

2.2.2.3 The Cellular Model

Figure 2.9 displays the cellular model from [Gong et al., 2015]. In the cellular model the population is distributed in a grid of cells where each cell holds one individual. Each individual only ”sees” the individuals of its neighborhood (as decided by the given neighborhood topology) and can only be compared with and mate with individuals in its neighborhood.

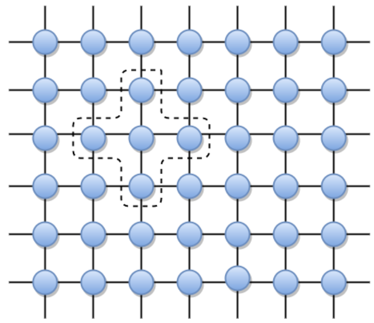


Figure 2.9: Cellular model where the neighborhood topology consist of the cells to the left, right, over and under the given cell [Gong et al., 2015].

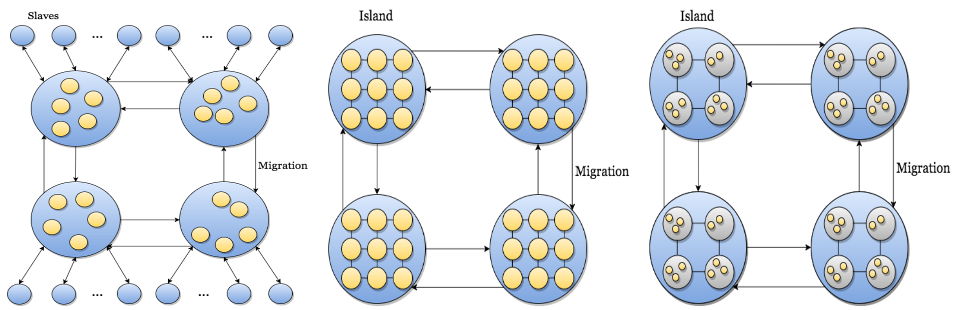
The takeover time is defined as the time it takes for one individual to propagate to the whole population. The neighborhood topology largely affects the takeover time. In figure 2.9 the neighborhood topology is defined as only the individuals left, right, over and under the given individual. Since the topology includes a small number of individuals the takeover time will be very long, meaning that exploration is prioritized. If the topology consist of a larger number of cells the takeover time will, off course, be mush shorter.

The cellular model can also be implemented in parallel, ideally with one processor for each cell. As in the island model, updating of the cells can be both synchronous and asynchronous. Synchronous updating means that all the cells are updated at the same time, while as asynchronous updating means that each cell is updated at a certain time following some updating scheme.

2.2.2.4 Hybrid Models

Hybrid models, are distributed genetic algorithms that combine different distributed genetic algorithm models. Gong et al. presented the three different hybrid models presented in figure 2.10.

The first hybrid model combines the island model with the master-slave model. By combining these two models, each deme will be processed faster



(a) (b) (c)

Figure 2.10: Different hybrid models. (a) Island-master-slave hybrid model, (b) Island-cellular hybrid model, and (c) Island-island hybrid model [Gong et al., 2015].

because it is distributed between different processors. The second model combines the island model and the cellular model. By combining these two models the diversity within a given deme can be kept longer than when the simple island model is used, and make sure premature convergence will not happen in the demes. The last model has a similar function as the second one, but instead of using the cellular model in each deme, it uses the island model inside the demes.

2.2.2.5 Pool Model

Another distributed model is called the pool model. In this model the population is put in a shared global pool of n individuals, where it can be accessed by different processors. Each processor draws a population from random positions of the pool, however it has allocated its own positions for which it can return individuals to the pool. This process is demonstrated in figure 2.11.

A processor p_1 has k positions in the pool for which it is responsible for as can be seen in figure 2.11, where the processor and the positions it is responsible for has the same color. It draws a population of individuals $i_1, i_2, ..., i_k$ from random positions in the pool, performs genetic operations and fitness calculations, and write each individual back to its corresponding position $1, 2, ..., k$ in the the positions it is responsible for, given that their fitness is higher then the fitness of the individual currently occupying the position. Key design de-

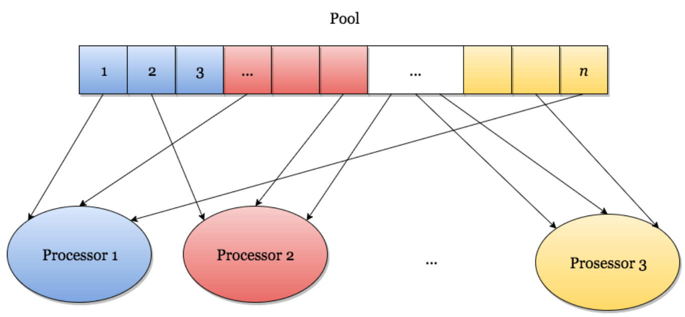


Figure 2.11: The pool model. Each processor has its own positions in the pool for which it is responsible to return individuals to. The red processor is responsible for the red positions in the pool, and so on. Processors draw individuals from random positions in the pool, but writes them back to its own positions, given that their fitness are higher than the individuals currently occupying the position [Gong et al., 2015].

cisions in this model is individual selection policy and replacement policy.

Chapter 3

Related Work

Wind farm layout optimization has been studied extensively the last 20 years and the goal of this section is to provide the reader with an overview. This section is divided into three parts; (1) An extensive overview of wind farm layout optimization using the genetic algorithm, since that is the main focus of this thesis, (2) an short review of other optimization approaches, and (3) a discussion that relates previous work to the current study.

3.1 Wind Farm Layout Optimization using Genetic Algorithms

Mosetti et al. [1994] were the first to successfully demonstrate the utilization of the genetic algorithm in solving the wind farm layout optimization problem. Although their work was made for illustrative purposes only, it laid the foundation for a number of more extensive studies of wind farm layout optimization using genetic algorithms.

In order to model a wind farm one have to specify a wake model, a power curve and a cost function. To model the wind decay, Mosetti et al. used a wake model similar to the one developed by Jensen [1983]. Power generated by each turbine i was modeled as a cubic function of the wind speed u and site roughness z_0, and summarized to get the total power produced by the farm in one year as shown in equation 3.1. Cost was modeled as a simple function of the number of turbines N_t, assuming a cost reduction when a large number of turbines are installed as shown in equation 3.2

$$Power_{total} = \sum_{i}^{N_t} z_0 u_i^3, \tag{3.1}$$

$$cost_{total} = N_t \left(\frac{2}{3} + \frac{1}{3} e^{-0.00174 N_t^2} \right). \tag{3.2}$$

With goal of producing a great amount of power at low cost, the objective function was formulated as a function of equation 3.1 and 3.2

$$Objective = \frac{1}{P_{total}} w_1 + \frac{cost_{total}}{P_{total}} w_2 \tag{3.3}$$

where w_1 and w_2 are weights. In the current study, w_1 was kept small so that the focus would be on lowest cost per energy produced.

Mosetti et al. divided the wind farm terrain into a 10×10 quadratic grid so that a wind turbine could be installed in the middle of each cell. The optimization problem would then be to find which cells wind turbines should be installed in, in order to maximize power production and minimize cost. With this representation, an individual of the genetic search could be represented as a binary string of length 100, where each index represents a cell in the grid, so that a value of 1 means that a wind turbine is installed in the corresponding cell, and a value of zero means that there is no wind turbine in the corresponding cell. The genetic algorithm used was a simple, single-population genetic search where the fittest individuals where selected for reproduction using crossover and mutation. The crossover operation was performed at random locations with probability $0.6 < P_c < 0.9$ and mutation was performed with probability $0.01 < P_m < 0.1$. Figure 3.1 illustrates how an individual represents a wind park for a wind farm partitioned into nine cells.

The model was tested on a single type of turbines in three different scenarios; (a) single wind direction, (b) multiple wind direction with constant intensity, and (c) multiple wind direction and intensity. For each scenario, the results was measured against random configurations of 50 turbines. In scenario one, the efficiency of the random configuration was 0.50, while the efficiency of the optimized solution was 0.95. In the second wind scenario,

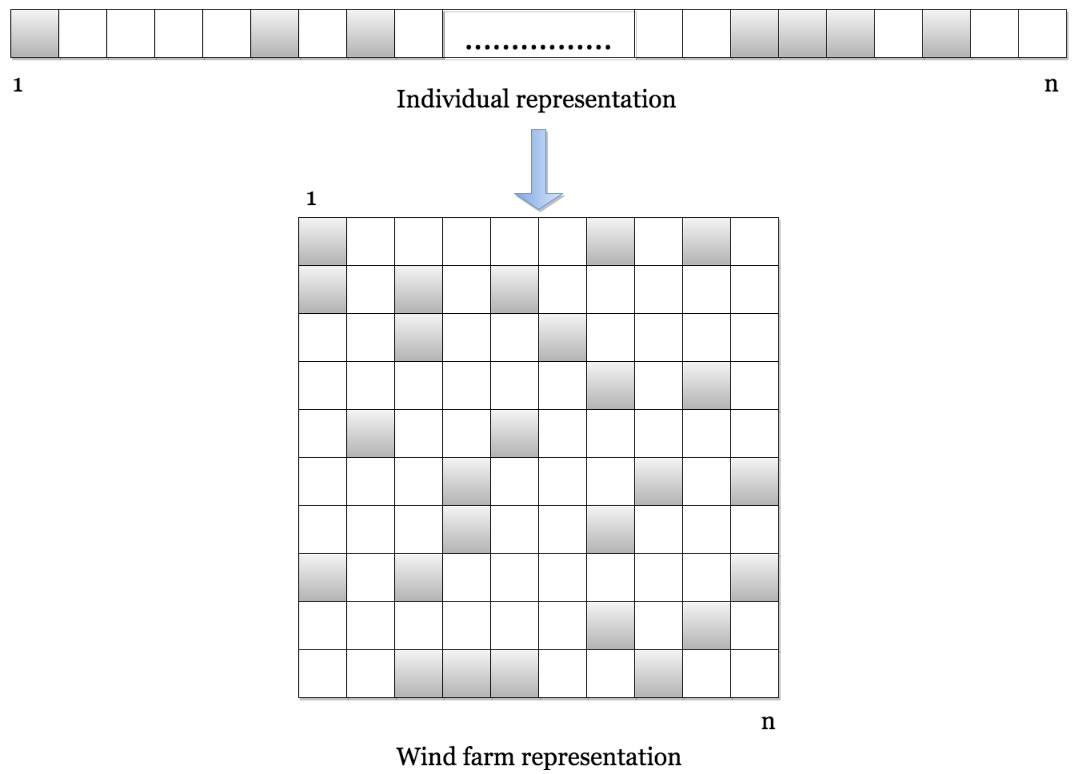


Figure 3.1: A simple example of how an individual of size nine represents a wind farm, where grey cells demonstrates positions containing a wind turbine and white cells demonstrates empty positions.

the efficiency was decreased from 0.35 in the random configuration to 0.88 in the optimized configuration. And, in the last scenario the efficiency was increased from 0.34 to 0.84. For each scenario the number of wind turbines was decreased drastically in the optimized version. Table 3.1 summarizes the results obtained.

As discussed in the paper, different simplifying assumptions are made in the model such as the wake effect model, the cost function, turbine type and layout model. The results are also only compared against random configurations, not configurations optimized by other optimization approaches, and

Table 3.1: Optimized configurations compared against random configurations for each of the three scenarios (a) single wind direction, (b) multiple wind direction with constant intensity and (c) multiple wind direction and intensity [Mosetti et al., 1994].

Scenario	Configuration	Efficiency	P_{tot}(kWyear)	cost/kWyear	Number of turbines
(a)	Random	0.50	13025	2.57×10^{-3}	50
	Optimized	0.95	12375	1.57×10^{-3}	25
(b)	Random	0.35	9117	3.68×10^{-3}	50
	Optimized	0.88	8711	1.84×10^{-3}	19
(c)	Random	0.34	4767	7.04×10^{-3}	50
	Optimized	0.84	3695	3.61×10^{-3}	15

no attempts is made to optimize the software. However, the purpose of this initial paper was to demonstrate the applicability of genetic algorithms on the wind farm layout optimization problem, and it has certainly laid the ground work for a number of studies performed over the last 20 years.

Grady et al. [2005] picked up where Mosetti et al. [1994] left of. They recognized that while the results of Mosetti et al. beat random configurations they were not close to configurations made on simple empirical placement schemes. In their study, they wanted to show that by implementing a distributed genetic algorithm the effectiveness of the algorithm could also be compared to optimal configurations. As Mosetti et al., they used the Jensen wake decay model, as well as the same cost- and power function. However, the objective function was changed into the following

$$Objective = \frac{cost}{P_{tot}}, \tag{3.4}$$

The same three scenarios as Mosetti et al. was considered. However, the number of individuals was increased from 200 to 600, and run for 3000 generations instead of 400. The distributed model used was an Island model where the individuals was divided into 20 sub-populations. Sadly, not many implementation details were shared. On the first scenario, Grady et al. recognized that with uniform wind distribution the optimal solution could be obtained by optimizing on single row of the layout, and copy it to the rest. As opposed to Mosetti et al., their results are identical to the optimal solution. In scenario (b) and (c) however, the optimal solution can not be obtained empirical, and therefore the results are just compared against those of Mosetti et al. The results for each scenario is displayed in table 3.2.

Table 3.2 compare the solutions of the two studies. The first thing to notice is the difference in number of turbines, where Grady et al. ends up with more turbines in each case, approximately doubling the number of turbines in scenario (b) and (c). The explanation behind this observation is the objective functions. Objective function 3.3 prioritizes low cost and hence prioritizes a lower turbine count. Fact check! The number of turbines for each case explains the largely explains the results. For each scenario the fitness of Mosetti et al. is higher than the fitness obtained in Grady et al. With exception of the first scenario, the efficiency is also larger in Mosetti et al., which makes sense since fewer turbines leads to less wake effect to decrease efficiency Fact check!. However, in each case, the total power production is largely increased in the current study, which also makes sense based on the turbine count Fact check!.

Table 3.2: Current results compared against the results from Grady et al. for each of the three scenarios [Grady et al., 2005].

Scenario	Parameter	Mosetti et al.	Grady et al.
(a)	Fitness	0.0016197	0.0015436
	Total power (kW year)	12 352	14 310
	Efficiency (%)	91.645	92.015
	Number of turbines	26	30
(b)	Fitness	0.0017371	0.0015666
	Total power (kW year)	9244.7	17220
	Efficiency (%)	93.859	85.174
	Number of turbines	19	39
(c)	Fitness	0.00099405	0.00080314
	Total power (kW year)	13 460	32 038
	Efficiency (%)	94.62	86.619
	Number of turbines	15	39

In summary, Grady et al. were able to show that by implementing a distributed genetic algorihtm optimal solution for scenario (a) can be obtained.

It also shows that the power production obtained in Mosetti et al. can be increased by optimizing of parameter values and more sophisticated implementation of the genetic algorithm. However, they make no attempt to compare their solutions for scenario (b) and (c) to solutions obtained using other optimization techniques. For a similar study where individuals are implemented as matrices in MATLAB see [Emami and Noghreh, 2010].

Zhao et al. [2006] presented a very interesting study, where the electrical system of an off shore wind farm on Burko Bank in Liverpool Bay was optimized using a genetic algorithm. Although this is a study of cable clustering design, with fixed wind turbine count and positions, it is very interesting because it is compared against an actual result obtained by the Burbo project team. To get an understanding of the optimization problem four different clustering designs are presented in figure 3.2.

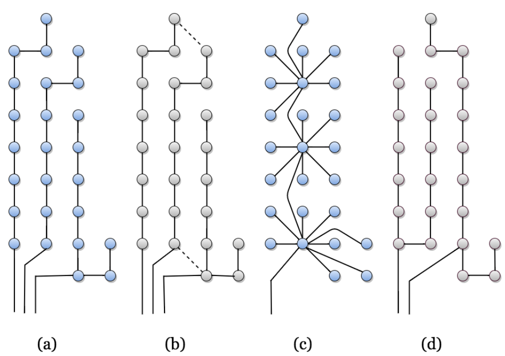


Figure 3.2: Four example clustering designs [Zhao et al., 2006].

In their paper, Zhao et al. presents an extensive study of different genetic algorithm techniques to find out which performs best on this type of optimization problem. Premature convergence is discovered as the main problem of the genetic algorithm and to deal with this different techniques are presented such as a diversity check, and an crowding technique called restricted tournament selection. For more implementation details suggest reading the paper? Can I write something like that.

Different genetic algorithm designs was tested, and the results show that the final design obtained was equal to the design obtain by the Burbo project team! This shows that optimization using sophisticated genetic algorithm implementations can find the same solution as current optimization techniques for optimization of electrical systems.

Huang [2007] presented a study, showing that a distributed genetic algorithm performs better than a simple genetic algorithm. Huang uses a more realistic objective function than the previous studies, taking into account the selling price of electric energy, as well as cost and energy production. The distributed genetic algorithm uses the Island model, with 600 individuals divided among 20 demes, using the ring-topology shown in figure 3.3.

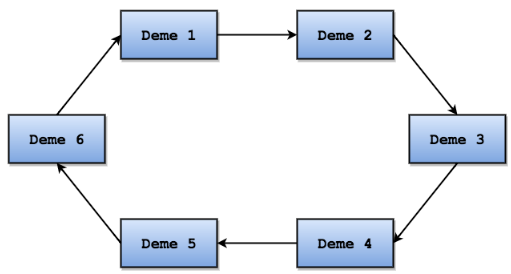


Figure 3.3: Ring topology example with 6 demes [Huang, 2007].

The simulation was run for 2500 generations with the migration strategy that 3.3% of the individuals with highest fitness was selected as migrants, to replace the individuals with lowest fitness in the new population every 20th generation. The distributed algorithm was tested using the same three scenarios as Mosetti et al. and Grady et al., against a simple genetic algorithm. In case (a) the distributed genetic algorithm was able to come up with the optimal solution (presented by Grady et al.), while as the simple genetic algorithm was not. For each of the three scenarios the distributed algorithm ended up with higher fitness value, more power produced, lower CPU time and fewer generations. In case (a) the turbine count was equal resulting in higher efficiency for the distributed algorithm, while in scenario (b) and (c) the distributed algorithm produced solutions with one more turbine than the simple algorithm, resulting in slightly lower efficiency. Huang also briefly gives a thin explanation that the results obtained are slightly better than those of Grady et al., but acknowledges that the results are hard to compare because of different objective functions.

All studies presented above have used binary encoding in their wind farm representation, but in 2007 Mora et al. presented a study where the binary encoding was replaced by an integer encoding [Mora et al., 2007]. In their approach, an individual was represented with the (x, y) coordinates of a given turbine. In addition to optimize turbine position, Mora et al. also wanted to optimize both turbine type and height. In order to do this, their individuals was represented by a matrix where the first row contained the x-coordinates of the turbines, the second row the y-coordinates, the third row turbine type and the forth row turbine height as shown in figure 3.4. Note that with this type of encoding, different individuals can have different lengths, depending on the number of turbines in each solution. Whether it is reasonable to allow different turbine types is another discussion, which will not be taken here, but previous authors have mentioned that when the same turbine type is used one can expect the price of a turbine to decrease with the number of turbines purchased Reference Mosetti and the others, but double check first.

	Turbine 1	Turbine 2		Turbine k
X-Coordinate	X_1	X_2	...	X_k
Y-Coordinate	Y_1	Y_2	...	Y_k
Turbine type	T_1	T_2	...	T_k
Turbine height	H_1	H_2	...	H_k

Figure 3.4: Representation of an individual of length k (layout with k turbines), where the first row represents x-coordinates of the turbines, the second row y-coordinates, the third row turbine type, and the forth row turbine height [Mora et al., 2007].

Five crossover methods, and a masked mutation method were presented for the new type of encoding, see Mora et al. [2007] for implementation details. To model the wind speed, the Weibull distribution was used, a more realistic wind speed model than the one used in the previous studies. The Weibull distribution will be explained in more detail in section reference section Weibull, since it is also used in the simulator for this thesis. Three different case studies are performed, the first one searches for an optimal solution when the number of turbines are decided beforehand. The second searches for an optimal positioning, type and height of turbines within a given budget. And the third one where there are no such constraints. Even thought the results are only briefly discussed, and presented by the authors as optimal with no explanation, this paper marks the switch from binary encoding to integer encoding, and from simple wind models to the Weibull distribution.

Wan et al. [2009] criticized the simple power-, and wind distribution model presented by Mosetti et al., and Grady et al. As Mora et al., they suggested using the Weibull distribution to model the wind. However, they introduced a novel power model

$$\int_{u_{in}}^{u_{out}} P(u)f(u)du, \tag{3.5}$$

where u_{in} is the cut-in wind speed of the turbine, and u_{out} is the cut-out wind speed of the turbine. $P(u)$ is the power output for the wind speed u and $f(u)$ is the probability density of the wind speed u. The genetic algorithm was similar to that of Grady et al., and results show that the produced power increases. The results are not that important thought, because different objective function will produce different optimal solutions, however the power model introduced will be utilized in many future studies.

One of the most complete studies of the wind farm layout optimization problem was done by Kusiak and Song [2010]. Should I write et al. when only two authors? The study is based on six assumptions, which according to the authors are realistic and industrial-accepted. The study assume a fixed, predetermined turbine count, small variations of surface roughness, turbines with equal power curves, wind speed following the Weibull distribution, that wind speed at different locations with same direction share the same Weibull distribution, and last, it assumes that any two turbines are separated with at least four rotor diameters. A multi-objective function was used to calculate the fitness of the solutions. It consisted of one objective function to maximize expected energy produced, and one to minimize the constraint violations. Kusiak et al. critiques Mosetti et al. and Grady et al. for not basing their wind energy calculation on the power curve function and not thoroughly discussing wind direction. Their work include an extensive model of wind energy based on a discretization of the expected power production for each wind direction. Their algorithm was tested on real wind data, and compared against an upper bound on power production (power produced without wake effect), and their results show that less than 2% of power is lost due to wake effects when 6 turbines are positioned in the wind farm.

Assumptions such as a cost model only dependent on the number of turbines are unrealistic, and needs to be removed in order to model the wind farm layout optimization problem in an realistic way. González et al. [2010] introduced a cost model based on the net present value, which takes into account wind speed, wind distribution, the number, type, rated power and tower height of turbines, loss to due wake effects, auxiliary costs, road infrastructure, buildings, substation, electrical framework and financial aspects such as return on investment. For example, In order to accurately model civil cost they present an greedy search which connects wind turbines to auxiliary roads or other turbines dependent on their position. Figure 3.5 shows how the greedy algorithm works. To connect the first turbines to one of the roads it measures the distance between every turbine and every road and connects the turbine and road that are nearest. Next it calculates the distance between the remaining turbines and the two roads and the turbine already connected, finds the minimum distance and connect the turbines. Not done. Explain algorithm using example. Make caption.
Other new features included is a new wake model Frandsen et al., 2006, 2007, a local search when the genetic algorithm cannot find a better individual, and a genetic algorithm that can manage forbidden areas and that gives penalties for turbines positioned in undesirable terrain. Individuals are represented the same way as Mora et al. [2007] displayed in figure 3.4. Results are compared against Grady, and shows higher produced energy. The authors also include three case studies showing how their algorithm can handle restrictions such as roads crossing, forbidden zones, undesirable zones and maximum investment cost.

Şişbot et al. [2010] published a case study of wind turbine placement on a wind farm at the Island Gökçeada, at the north east of the Aegean Sea. A distributed genetic algorithm was used, but, unlike Huang, the individuals were evaluated based on multiple objective functions; one that measures the total cost (installation and operational), and one that measure total power production. Şişbot et al. argue that in an environment with changing demands, the use of a multi-objective function gives the decision-makers the

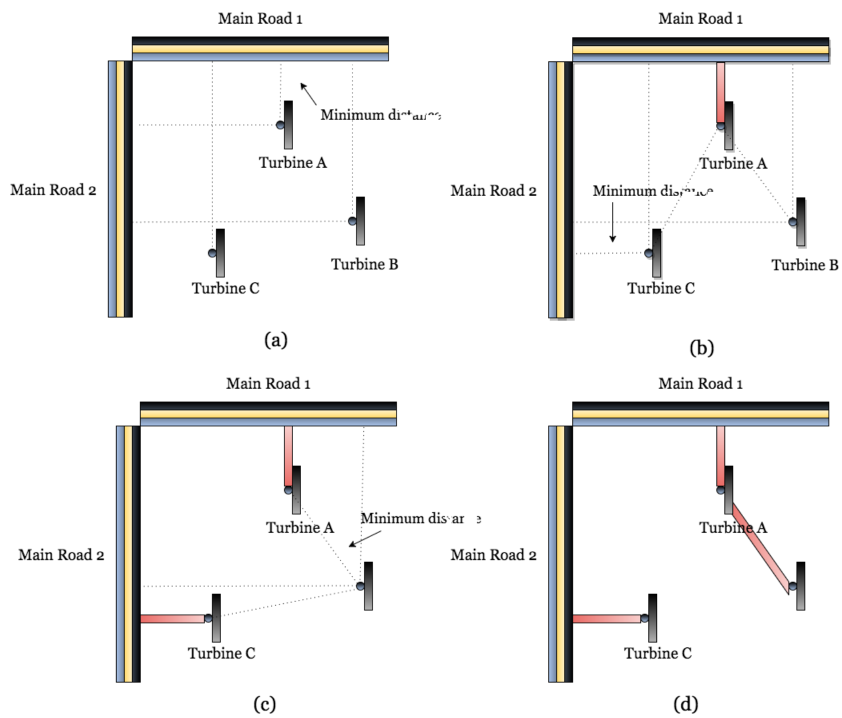


Figure 3.5: Add caption [González et al., 2010].

opportunity to evaluate the different designs based on cost and power production separately, without ill-informed, randomly generated weights. The selection process used is a controlled, elitist process, meaning that not only the fittest, but also some individuals that can spread diversity to the population are selected for reproduction. The genetic algorithm returns a set of Pareto optimal solutions; a set of solutions that are not dominated by any other solution in the set. Stated more formally, solutions $\mathbf{y}$ is said to dominate solution $\mathbf{x}$ if

$$\forall i : f_i(\mathbf{x}) \leq f_i(\mathbf{y}) \; and \; \exists j : f_j(\mathbf{x}) < f_j(\mathbf{y}) \tag{3.6}$$

where f_i is objective function [Murata et al., 2001]. Other interesting features of this study is the introductions of constraints on wind turbine positions and constraints on the cost, meaning that individuals with wind turbines outside the area of the island, and individuals with costs larger than the budget are removed from the population. Even thought constraints on individuals are not in accordance with the nature genetic algorithms, they can be necessary when the algorithm is applied to a real problem. Another feature introduced in this paper is rectangular cells. The argument behind this decision is that the safe distance between wind turbines is dependent of the direction of the turbine. The minimum distance between turbines in prevailing wind is 8D, while the minimum distance between turbines in the crosswind is 2D. In spite of this attempt to make the wind scenario more realistic, it is critiqued because it operates with a constant wind direction and constant speed, using the average wind direction and speed measured at the Island [Samorani, 2013]. Results are not compared against previous studies, and the argument behind this decision is that it is hard to compare a Pareto-optimal set of solution to one of the previous solutions.

Another very interesting solution to the wind farm layout optimization problem was proposed by [Saavedra-Morena et al., 2011]. Four novel improvements were included in their model. First, a shape model was introduced to model the terrain shape. By introducing this model, the simplification of a square grid was lost, and every terrain shape could be implemented. Second, an orography model was used to model the wind speed on different height differences. Using this technique, the wind model is much more realistic because it takes into account that wind speed differs at different heights. Third, they introduce a new cost model, which takes into account installation cost, connection between turbines, road construction and net benefit from the produced energy. The fourth, and maybe most exiting improvement has been discussed and requested by Mosetti et al. [1994]. Instead of starting the genetic search from a random population, a greedy-constructive heuristic is used to decide the initial positioning of the turbines for some of the individuals. The greedy heuristic works by placing turbines one by one in the position with maximal wind speed. First, the first turbine is positioned in the position wind maximum wind speed, next the wind speed is updated because of the reduction in wind speed caused by the wake effect of the first turbine, third, the second turbine is positioned in the position with maximal wind speed. This process continues until N wind turbines is placed. Clearly, the resulting layout is largely influenced by the positioning of the first turbine, and leads to a sub-optimal solution on its own. However, it is much better than a random solution, and as it turns out a good starting point for the genetic algorithm. Results for 15 different orographys for the same terrain shape is provided, showing the objective function values obtained by the greedy heuristic, a simple genetic algorithm with random starting positions, and the seeded genetic algorithm (the genetic algorithm with starting positions provided by the greedy heuristic). In each case, the genetic algorithm with random starting positions beats the results obtained by the greedy heuristic alone, but more importantly, in each case, the seeded genetic algorithm beats the results of the simple genetic algorithm.

Both Mora et al. [2007] and González et al. [2010] used the genetic algorithm to optimize the height of the turbines, as well as other parameters by representing individuals as shown in 3.4. Another approach to optimize turbine height was presented by Chen et al. [2013]. They state that normally, the same turbine type can be bought with several different heights, and that it therefore makes sense to use different height turbines. To optimize turbine position and height, they used two nested genetic algorithms. They first one was used to optimize turbine positioning, while the second one was used to decide between two turbine heights. For each generation of the first genetic algorithm, the second one where run for 50 generations to optimized turbine height. Binary encoding was used for individual representation in both genetic algorithms as shown in figure 3.6. The first binary string represent turbine positioning in the environment, while the second binary string represents turbine height for each position that contains a turbine.

Several case studies were performed in the paper, and results show that turbine layout with different turbine height, produce more energy than same-height turbines every time.

In 2005, Gao et al. implemented a distributed genetic algorithm to solve the wind farm layout optimization problem. Unfortunately, they have kept most of the implementation details of the distributed genetic algorithm for themselves. Integer encoding are used to represent individuals, but unlike those presented earlier list everyone, each individual have the same number of turbines. The algorithm is tested on the same three scenarios from Mosetti et al. and compared against all previous studies using the same scenarios [Mosetti et al., 1994], [Grady et al., 2005], [Gonzàlez et al., 2010], [Mittal, 2010], [Pookpunt et al., 2013,][Wan et al., 2009, 2010], [Zhang et al., 2011]. For the comparison to be valid they force their solutions to have the same number of turbines as the results of the previous studies. In each case their resulting layout is able to produce more energy, and has higher efficiency,

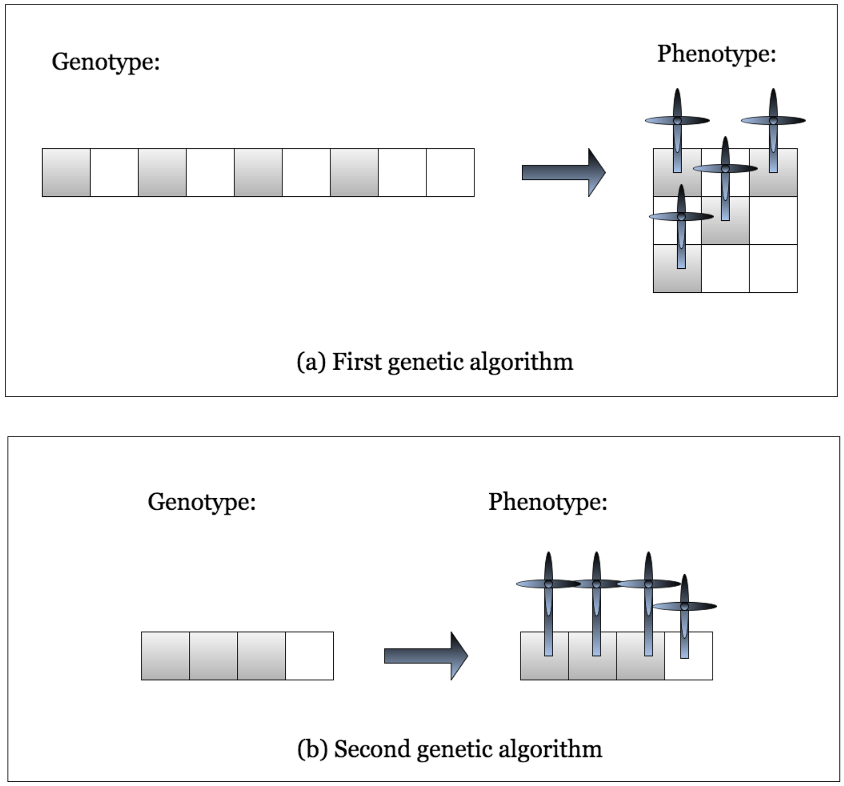


Figure 3.6: Representation for both genetic algorithms from [Chen et al., 2013] Chen et al., (2010). (a) Binary string representing turbine positions, (b) binary string representing height of the given turbines positioned by the first genetic algorithm.

while never achieving the highest fitness. In addition to this comparison, Gao et al. introduce an interesting hypothetical case study of wind turbine placement on an offshore farm located in the Hong Kong southeastern water. By using real wind data, collected over 20 years, they demonstrates that the distributed genetic algorithm can be applied to a real-world wind farm layout optimization problem. The resulting wind farm layout was able to produce 9.1% of yearly electrical consumption in Hong Kong (2012). Should I remove reference to studies that I dont use?

3.2 Wind Farm Layout Optimization Different Approaches

A greedy heuristic approach to the wind farm layout optimization problem was presented by Ozturk and Norman [2004]. The algorithm starts out with an initial solution, where a number of wind turbines are positioned in the wind farm. Next, the greedy algorithm tries to improve the layout by performing either an add operation, a remove operation or a move operations. The add operation works by randomly position one new turbine in the terrain a number of times at different locations, and observe the change in the objective function. The remove operation works by observing the change in objective function when a turbine is removed, the process is repeated for all turbines. A move operation consist of moving each turbine 4 rotor diameters away from its current position in eight wind direction on at a time, and observe the change in the objective function. The operation actually performed by the algorithm is the one that improves the objective function value the most. The greedy heuristic often converge to a local optimum, and the authors try to cope with this problem by performing randomly perturbations on a number of turbine positions if the no improvements can be found using the add-, remove-, or move operation. Three approaches was investigated to find the initial position of the turbines; (1) randomly positioning, (2) packing the wind farm with as many turbines as possible, and (3) start with zero turbines. Preliminary testing showed that the second approach produced best results. The greedy algorithm was tested, and the results were compared to a feasible solution with the maximum number of turbines positioned, i.e. the initial layout before the algorithm is run. In 10 out of 12 case studies the algorithm improved the layout of the wind farm, with an average improvement of 4.3%.

Marmidis et al. [2008] used Monte Carlo simulation to find the optimal positioning of turbines. They used the same wake model, power curve and cost model as [Mosetti et al., 1994] so that they could compare results. However, they only tested their algorithm on the first scenario of uniform wind speed. The simulation works as follows; Random solutions are generated and compared against previous optimal solutions. If the newly generated solutions are better than the current optimal solutions they are stored. Results using Monte Carlo simulation obtains better fitness value and higher produced energy, however, it should be compared to other scenarios, since the first wind scenario is extremely unrealistic. How can they beat the optimal result? Don't get how the Monte Carlo simulation works.

Bilbao and Alba [2009] designed a simulated annealing algorithm to solve the wind farm layout optimization problem. The same wind parameters and representation as Grady et al. [2005] was used, and the algorithm was tested on the same three scenarios. The simulated annealing algorithm works as follows; first, an initial layout is obtained by randomly position a predefined number of turbines. Later, a random position that contains a turbine is chosen, and a new, randomly generated location is suggested. If the new position is better, the turbine is moved, but if the new position is not better, the turbine is moved with a certain probability which is regulated by a decreasing temperature parameter, in order of preventing the algorithm of converging to a local optimum instead of the global optimum. In case (a) of [Grady et al., 2005] the simulated annealing algorithm is able to find the same optimal solution, and that by using only $\approx 4\%$ of the execution time of Grady et al., and only $\approx 1\%$ of the time spent evaluating the solution. In case (b) and (c) the simulated annealing algorithm is able to find solutions with better fitness, higher power production, higher efficiency, and significantly lower execution- and evaluation times, showing that simulation annealing might be a good technique to search for the optimal wind farm layout, and it should definitively be tested in a more realistic environment.

The ant colony algorithm, is an algorithm that is inspired by how ants search for food, and show other ant food sources based on leaving a pheromone trail. Eroğlu and Seçkiner [2012] proposed an ant colony algorithm for the wind farm layout optimization problem. The algorithm operates on a predetermined number of turbines, randomly positioned. The pheromone quantity of each turbine is decided by wake loss for the given turbine, resulting in a stronger pheromone trail for turbines with worse locations. Ants will follow the pheromone trail, therefore more ants will try to better the position of the worse turbines by moving them in random directions - the turbine is only moved if the new position is better than the current. Results are compared against [Kusiak and Song, 2010] and it is shown that the ant colony algorithm was able to position two more turbines; eight turbines in total, and that when the number of turbines is greater than two, the current algorithm produce more power, has less wake loss and higher efficiency.

Another technique to search for the best wind farm layout is swarm optimization, and Wan et al. [2012] demonstrates how a Gaussian particle swarm algorithm can solve the wind farm layout optimization problem. Swarm optimization, is an optimization technique inspired by fish schooling, insect swarms and bird flocking. The algorithm presented used an objective function that tries to maximize produced power, while minimizing constraint violations. The algorithm works as follows: First, N particles are placed in random (x, y)-positions. Second, the initial solution is evaluated and the results are saved. Third, the population best position z^g is saved, along with the current best position observed for each particle Z^p, which in the beginning will be the initial positions. An algorithm is presented, to decide which, out of two layouts, it the best. It works by first prioritizing layouts which violates less constraints, and second compare power produced by the two layouts. Forth, an updating scheme is run for a given number of iterations. It first checks if the local best position observed by that particle z^p is equal to the global best position z^g, and if so, uses an regeneration scheme that moves the particle to a random position. Otherwise, a new position is calculated for the particle based on the particles current position, its current best observed position and the global best observed position and two normalized random Gaussian numbers, if the new position is better than the previous one, the particle is moved. A differential evolution local scheme is also incorporated in each iteration to improve the algorithms local search ability. It basically work by randomly picking three random particles as potential parents for a given particle, and combine these to generate an alternative new position, which is assign to the particle, if it is better than the current one. Results were compared against [Grady et al., 2005], and show that the power generated is higher using this algorithm. Their algorithm is also tested in a more realistic environment and compared against an empirical method as well as a simpler particle swarm algorithm, and shows that the power generated is increased using the proposed algorithm.

3.3 Discussion Related Work

In order to provide an overview of the different publications presented in the two previous sections the wake-, wind-, power-, and cost model, along with objective functions, type of genetic algorithms and a short explanation of novelties presented in each article is presented in table **??**.

As can be seen in the table, each article utilized some variation of the Jensen model, developed by [Jensen, 1983], and later improved upon by [Katic et al., 1986] and [Frandsen et al., 2007]. The same model will also be used in this thesis.

Even though not many improvements has been made to the wake model, the wind model has evolved a lot since [Mosetti et al., 1994]. As presented before, the three wind scenarios developed by Mosetti et al.; (a) single wind direction, uniform intensity, (b) multiple wind direction, uniform intensity, and (c) multiple wind directions and intensity, are not very realistic. The Weibull distribution, introduced by [Mora et al., 2007], models the wind distribution much better, and has been adopted by everyone, except those who still wanted to compare their results against [Mosetti et al., 1994] and [Grady et al., 2005]. As mentioned before, the Weibull distribution will also be used to model wind distribution in this thesis, and will be described in more detail in section Reference section, when the simulator is explained.

The majority of the publications presented uses the simple power model presented by [Mosetti et al., 1994], however, [Kusiak and Song, 2010] presents a more complex, linear power model, which also will be used in this thesis and are explain in section Reference section.

The quality of the cost model has varied greatly in the different studies. The very unrealistic cost model that only takes turbine count into account has been adopted by many, as can be seen in table **??**. However, [Mora et al., 2007], [**?**], Sisbot, [Saavedra-Morena et al., 2011], and [Chen et al., 2013] used more realist cost models, taking into account different parameters such as net present value, installation cost, maintenance work, civil work, interest rate and so on. In the current thesis, a very complicated objective function is presented, one which takes into account turbine cost, substation cost, interest rate, operating costs, and turbine count as well as produced power. Section Reference section gives an detailed explanation of the objective function.

In this thesis, the wake-, wind-, power-, and cost model, and objective function was provided by GECCO 2015, and therefore the main focus will be on improving the layout using the genetic algorithm. Since the simple genetic algorithm presented by [Mosetti et al., 1994], different approaches has been tested in order to improve the results. Already in 2005, it was shown that by using a distributed genetic algorithm, and change a few parameters, the results from [Mosetti et al., 1994] was improved [Grady et al., 2005]. In [Huang, 2007] the focus was on showing how distributed genetic algorithms performs better than simple genetic algorithms on solving the wind farm layout optimization problem, and his results show that the distributed genetic algorithm is never beaten by the simple genetic algorithm. Both Grady et al., and Huang et al., uses the Island model when implementing distributed genetic algorithms. Even though these results indicate the distributed genetic algorithms works better, the results of Grady et al., is beaten by the simple genetic algorithm of **?** when a local search is used together with the genetic algorithm. Also, in [Saavedra-Morena et al., 2011] it is shown that a seeded simple genetic algorithm shows promising results, even though it is not compared against a distributed genetic algorithm. [Gao et al., 2014] also demonstrates how their distributed genetic algorithm performs better than [Mosetti et al., 1994], [Grady et al., 2005], [Gonzàlez et al., 2010],[Wan et al., 2009, 2010], and [Zhang et al., 2011], but sadly they do not share many implementation details. These results clearly show that distributed genetic algorithms can be an effective optimization technique for the wind farm layout optimization problem and they will therefore be the focus of this thesis. Together with the Island model, all six distributed algorithms presented in Reference section will be implemented and tested in this thesis. The goal is to find out which one is the most useful approach for the wind farm layout domain.

Paper	Wake Model	Wind Model	Power Model	Cost Model
Mosetti (1994)	Jensen inspired	(a) Single wind direction,mean wind speed (8m/s) b) Multi-directional Wind (uniform distributed) (12 m/s), c) uniform + gaussian centered at 270-350 (17m/s)	Betz	Mosetti
Grady (2005)	Mosetti	(a) Wind intensity and direction fixed b) Multi-directional Wind (uniform distributed) c) uniform + gaussian centered at 270-350	Betz (Mosetti)	Mosetti
Huang (2007)	Katic		Mosetti	Mosetti
Mora (2007)	NA	a) W(K=3,C=8,14) b) W(K=5,C=8,14) c) W(K=[3,5],C=0,15)	NA	Complex
Emami (2009)	Jensen,Simplified	a) Wind intensity and direction fixed b) Wind intensity fixed, direction uniformly distributed	Mosetti	Mosetti
Wan (2009)	Jensen, Mosetti	a) 1 wind direction, Speed weibull b) multiple wind directions, speed K=2,C=13.54	Mosetti	Mosetti
Kusiak (2009)	Mosetti	a) K=2, C=13, b) K=2, C=2.6-10	linear a=140.86,b=-500 ,Prated=1500,Vcin=3.5,Vcout=20,Vrated=14	NA
Gonzales (2010)	Frandsen	K=2, C=11.19, 9.51, 9.51, 8.39	Mosetti	Operational and installation costs
Sisbot (2010)	Jensen, Mosetti, Katic, Grady	Single Direction, Constant Speed 487.8u^3	Betz	installation and operational costs
Saavedra-Moreno	Mosetti	Weibull(K=,C=),*	Betz (eq 8)	Turbine installation and infrastructure M
Chen (2013)	Jensen, Katic, Frandsen	(a) Constant, (b) const. speed, semi-uniform direction, c) varying speed and direction fig 8 (equal shapes)	Betz	third case (1) Mosetti, third case (2) Mor
Gao et al (2014)	Jensen, Mosetti	(a) Constant, (b) uniform (c) Histogram, real data	Betz	Mosetti

Chapter 4

Technical Description

Since this thesis is part of the Genetic and Evolutionary Computation Conference 2015, the conference has provided all the contestants with an API available on a GitHub. API's for different programming languages are provided, but the Java API will be used in this thesis. The API includes an implementation of a simple genetic algorithm, an evaluation method and ten different wind scenarios. This section will give an explanation of the provided API, display the results from a few test simulations (which will not be included in the final thesis), and discuss future work.

4.1 Application User Interface (API)

A class diagram of the provided API is displayed in figure 4.1. As can be seen, the diagram consist of six classes. The GA.java class which contains a simple genetic algorithm, i.e. the whole process displayed in figure Want to reference the first figure of the genetic algorithm process. Layout evaluation is taken care of by the evaluation classes. WindFarmLayoutEvaluator.java is an abstract class, implemented by KusiakEvaluator.java and CompetitionEvaluator.java, where the last one connects the evaluation cost function of the competition to the online server of the competition. Next, the class called WindScenario.java is used to initialized one of the ten wind scenarios provided by the competition. Running the program is simple, it only requires initializing the wind scenario, layout evaluator, and the genetic algorithm, and then start the run() method in the GA.java class.

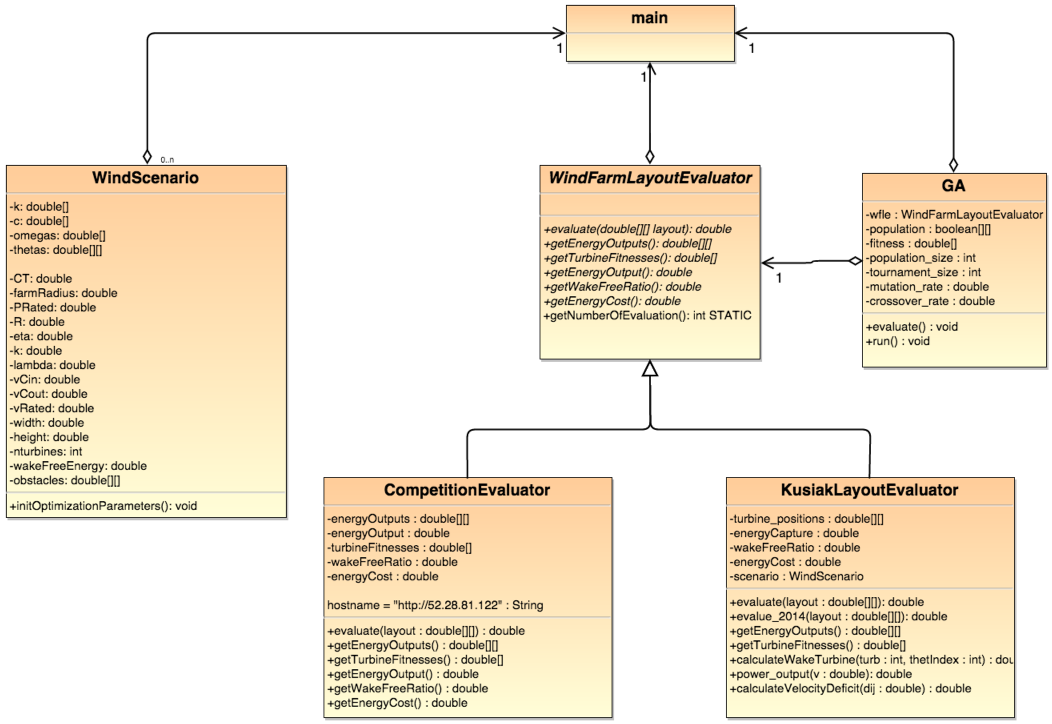


Figure 4.1: Class diagram of the API provided by GECCO 2015.

4.1.1 Genetic Algorithm

The genetic algorithm provided is a simple genetic algorithm that uses a binary representation. Even thought a binary representation is used, it is slightly different from the approaches listed in related work, because positions that are not allowed is removed from the representation. This is shown in figure 4.2. Two of the nine positions, the red ones, contains obstacles and are therefore not allowed, these are therefore removed from the individual. This means that even though the wind farm is partitioned into 9 squares, the individual is only represented with a binary string of size seven, since there is only seven legal positions. The selection method, and crossover and mutation methods are the ones presented in the section Reference background section.

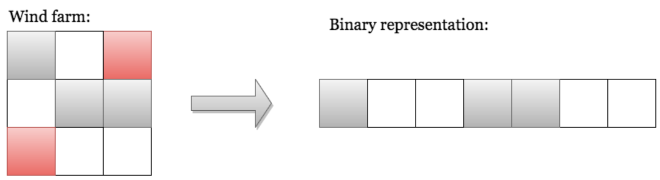


Figure 4.2: Binary representation of a wind farm. Red cells contains obstacles and are not included in the binary representation. White cells are empty, grey cells contains turbines.

4.1.2 Fitness function

The main task of the evaluation classes are to calculate the fitness of each individual based on the objective function. The objective function to be optimize is provided by GECCO, and is displayed in equation 4.1.

$$fitness = \frac{\left(c_t \cdot n + c_s \cdot \left\lfloor \frac{n}{m} \right\rfloor\right)\left(\frac{2}{3} + \frac{1}{3} \cdot e^{-0.00174n^2}\right) + c_{OM} \cdot n}{\left(\frac{1-(1+r)^{-y}}{r}\right)} \cdot \frac{1}{8760 \cdot P} + \frac{0.1}{n} \tag{4.1}$$

Description and value of all parameters given in equation 4.1 are displayed in table 4.1. As can be seen in this table, the values of n, the number of turbines, and P, farm energy output, are not given. This is because the number of turbines, together with the turbine positions, are the parameters to be optimized by the genetic algorithm. Farm energy output is the indirect parameter that we are trying to optimize, it is dependent on turbines count, position, wind scenario and so on, and is of course therefore not provided in table 4.1 either. Explain fitness well! What is positive, negative.

Intuitively, the objective function can be divided into different parts. The first parenthesis in the nominator of the first fraction is the construction cost, while the second parenthesis is the economies of scale and the third part of the nominator is yearly operating costs. The denominator is the interests. The denominator of the second fraction describes yearly power output, while the number 0.1 in the nominator of the last fraction is a farm size coefficient. Not finished with this section. Don't know if wake, power curve and stuff

Table 4.1: Description and value of each parameter used in the objective function provided by GECCO 2015.

Parameter	Description	Value
c_t	Turbine cost (usd)	750 000
c_s	Substation cost (usd)	8 000 000
m	Turbines per substation	30
r	Interest rate	0.03
y	Farm lifetime (years)	20
c_{OM}	Yearly operating costs per turbine	20 000
n	Number of turbines	
m	Farm energy output	

should be here. Explain how power is calculated, since it is needed to calculate fitness. Explain how every term in the objective function is calculated, then this is done.

4.1.3 Wake-, Wind- and Power Model

The evaluation class uses the same wake-, wind- and power model as [Kusiak and Song, 2010]. The wake model used is the classical Jensen model [Jensen, 1983], which is used in almost every study of the wind farm layout optimization problem, as can be seen in table Reference table from RW.

Wind distribution is modeled using the Weibull distribution, a continuous probability distribution shown to model wind distribution quite well [Justus et al., 1978]. The probability density function is shown in equation 4.2

$$f(x; c, k) = \begin{cases} \frac{k}{c}\left(\frac{x}{c}\right)^{k-1} e^{-\left(\frac{x}{c}\right)^k} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{4.2}$$

where k is called the shape parameter and c is the scale parameter, and $k, c > 0$. In most of the wind scenarios provided by GECCO 2015, $k \approx 2$, this is shown empirically to be a good value for wind speed distribution [Justus et al., 1978]. On the other hand, the shape parameter vary for each wind direction. Figure 4.3 shows the Weibull distribution plotted for different values of k and c.

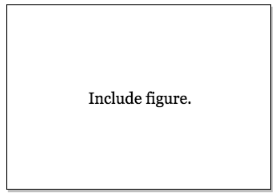


Figure 4.3: The Weibull distribution plotted for different values of k and c.

The wind scenarios used in this thesis is therefore an specification of the shape- and scale parameters for every wind direction, where wind direction is partitioned into 24 different directions. Twenty wind scenarios are provided be GECCO 2015, ten which simply specify wind distribution, and ten that specify wind direction and locations of obstacles.

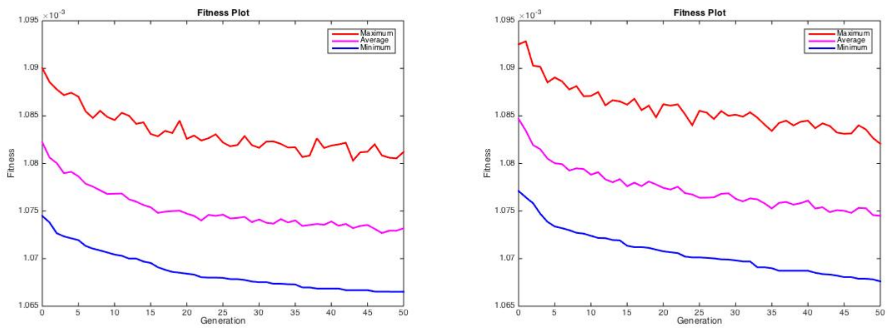
The power curve used is also the same as used in [Kusiak and Song, 2010], it is a linear function, shown in equation 4.3.

$$f(v) = \begin{cases} 0 & \text{if } v < v_{cut-in} \\ \lambda v + \eta & \text{if } v_{cut-in} \leq v \leq v_{rated} \\ P_{rated} & \text{if } v_{cut-out} > v > v_{rated} \end{cases} \tag{4.3}$$

where λ is the slope parameter, v the wind speed, η the intercept parameter, P_{rated} is the fixed power output, and v_{cut-in} is the cut-in speed; the minimum speed for which the turbine produces power, and $v_{cut-out}$ is the cut-out speed; the maximum wind speed for which the turbine is kept on.

4.2 Test Simulation

In order to demonstrate that the provided environment works, two test runs were performed. Figure 4.4 shows the resulting fitness plots when the provided genetic algorithm was ran once for scenario 1 without obstacles, and once for scenario 1 with obstacles, using default parameters. Each simulation was run for 50 generations, ten times, and the plots show the average fitness over these ten runs. As can be seen, the fitness decreases as the population becomes better and better solutions to the problem. The scenario without obstacles are simpler and converges therefore to a lower fitness value than the scenario with obstacles. Note that these simulations are only ran to show that the simulation works, and the results will not be included in the final thesis.



(a) (b)

Figure 4.4: Minimum-, maximum-, and average fitness for two scenarios provided by GECCO; (a) Fitness for scenario 1, without obstacles, (b) Fitness for scenario 1 with obstacles. Each simulation was run for 50 generations, ten times, and the result is averaged over these ten runs.

4.3 Future Work

In order to investigate the goal statement and answer the research questions the provided genetic algorithm needs to be extended as shown in figure 4.5. The provided genetic algorithm will be used as a super class, which will be extended by different distributed genetic algorithms, such as the master slave model, the island model, the cellular model and a few hybrid models as well. For details of the distributed models that will be implemented see section Reference background.

In addition to implementing the distributed genetic algorithms listed above, there might be a few changes done to the class GA.java depending on the "goodness" of the results, such as changes to the selection-, mutation and/or crossover methods. Other extensions might also be included if needed, such

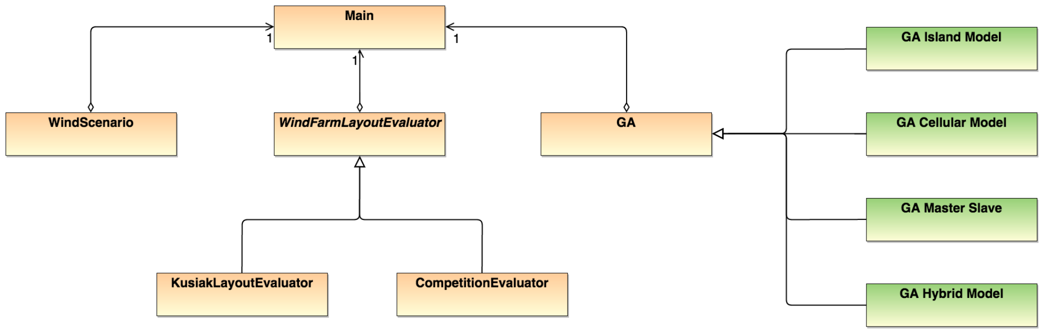


Figure 4.5: Simplified version of class diagram from figure 4.1 extended with different distributed genetic algorithms (green).

as seeding the genetic algorithm in order to get as good results as possible for the competition.

After the different distributed models are implemented, extensive simulations will be run for all the models, and results will be presented and discussed thoroughly.

Bibliography

Bilbao, M. and Alba, E. (2009). Simulated annealing for optimization of wind farm annual profit. In *Logistics and Industrial Informatics, 2009. LINDI 2009. 2nd International*, pages 1–5. IEEE.

Chen, Y., Li, H., Jin, K., and Song, Q. (2013). Wind farm layout optimization using genetic algorithm with different hub height wind turbines. *Energy Conversion and Management*.

Emami, A. and Noghreh, P. (2010). New approach on optimization in placement of wind turbines within wind farm by genetic algorithms. *Renewable Energy*.

Eroğlu, Y. and Seçkiner, S. U. (2012). Design of wind farm layout using ant colony algorithm. *Renewable Energy*, 44:53–62.

Frandsen, S. T. et al. (2007). *Turbulence and turbulence-generated structural loading in wind turbine clusters*. Risø National Laboratory.

Goldberg, D. E. (2005). *Genetic Algorithms in Search, Optimzation and Machine Learning*. Addison-Wesley Publishing Company.

Gong, Y.-J., Chen, W.-N., Zhan, Z.-H., Zhang, J., Li, Y., Zhang, Q., and Li, J.-J. (2015). Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*.

González, J. S., Rodriguez, A. G. G., Mora, J. C., Santos, J. R., and Payan, M. B. (2010). Optimization of wind farm turbines layout using an evolutive algorithm. *Renewable Energy*.

Grady, S. A., Hussaini, M. Y., and Abdullah, M. M. (2005). Placement of wind turbines using genetic algorithms. *Renewable Energy*.

Holland, J. H. (1992). *Adaptation in Natural And Artificial Systems*. The MIT Press.

Huang, H.-S. (2007). Distributed genetic algorithm for optimization of wind farm annual profits. *Intelligent Systems Applications to Power Systems*.

Jensen, N. O. (1983). *A note on wind generator interaction*. Technical Report Riso-M-2411.

Justus, C., Hargraves, W., Mikhail, A., and Graber, D. (1978). Methods for estimating wind speed frequency distributions. *Journal of applied meteorology*, 17(3):350–353.

Katic, I., Højstrup, J., and Jensen, N. (1986). *A simple model for cluster efficiency*. European Wind Energy Association Conference and Exhibition.

Kusiak, A. and Song, Z. (2010). Design of wind farm layout for maximum wind energy capture. *Renewable Energy*, 35(3):685–694.

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

Mora, J. C., Barón, J. M. C., Santos, J. M. R., and Payán, M. B. (2007). An evolutive algorithm for wind farm optimal design. *Neurocomputing*.

Mosetti, G., Poloni, C., and Diviacco, B. (1994). Optimization of wind turbine positioning in large windfarms by means of genetic algorithm. *Journal of Wind Engineering and Industrial Aerodynamics*.

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

Razali, N. M. and Geraghty, J. (2011). Genetic algorithm performance with different selection strategies in solving tsp.

Saavedra-Morena, B., Salcedo-Sanz, S., Paniagua-Tineo, A., Prieto, L., and Portilla-Figueras, A. (2011). Seeding evolutionary algorithms with heuristics for optimal wind turbines positioning in wind farms. *Renewable Energy*.

Samorani, M. (2013). *The Handbook of Wind Power Systems*, chapter The Wind Farm Layout Optimization Problem. Springer Berlin Heidelberg.

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

Wan, C., Wang, J., Yang, G., Gu, H., and Zhang, X. (2012). Wind farm micro-siting by gaussian particle swarm optimization with local search strategy. *Renewable Energy*, 48:276–286.

Wan, C., Wang, J., Yang, G., Li, X., and Zhang, X. (2009). Optimal micro-siting of wind turbines by genetic algorithms based on improved wind and turbine models. *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*.

Zhao, M., Cheng, Z., and Hjerrild, J. (2006). Analysis of the behaviour of genetic algorithms applied in optimization of electrical system designs for offshore wind farms. *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*.