Computing the Optimal Layout of a Wind Farm

Jan Kristian Haugland
Department of Informatics, University of Bergen, Norway
admin@neutreeko.net

Dag Haugland
Department of Informatics, University of Bergen, Norway
Dag.Haugland@ii.uib.no

Abstract

In this paper, we develop computational procedures for optimizing the location of turbines in a wind farm. We consider two different scenarios: First, we assume that the number of turbines to be installed is given, and the goal is to find their locations such that the total power generation is maximized. Every point within a defined region is a potential turbine location. Second, we assume a finite set of possible locations, typically defined in terms of a grid, and the goal is to determine at what locations a turbine should be installed. The goal in this scenario is to maximize the net profit constituted by future sales income minus turbine installation and running costs.

To reflect the interdependence between turbines, we need a model of the decline of the wind velocity behind a turbine. In the first part of the paper, we consequently develop a new wake model, which is shown to be an improvement of one of the most popular wake models in the literature.

In the second part of the paper, we suggest heuristic methods for approaching the optimization problems, and evaluate them experimentally on a number of test cases. Based on results from the experiments, we conclude that our new wake model, combined with the method displaying the best performance in the experiments, is a useful tool in designing wind farms.

1 Introduction

In a world of increasing demand for electrical energy, maximal production of coal and other fossil fuels possibly being reached within decades, and increased amount of extreme weather due to global warming, the use of wind turbines as a source of electrical energy is more topical than ever before.

A number of wind farms are already in existence [1]. Many of the world's largest onshore wind farms can be found in the United States, but also in other countries like

This paper was presented at the NIK-2012 conference; see http://www.nik.no/.

China and Australia. Smaller onshore wind farms can be found in a number of different countries all over the globe. The capacity of each farm can be anything up 720 MW (attained by the Alta Wind Energy Center in the U.S.).

The largest offshore wind farms are found in Europe, particularly in Denmark and Great Britain. The largest one is Walney off the coast of Cumbria in England, with a capacity of 367 MW. In Norway, Vestavind Offshore [2] is planning to build Havsul, a wind farm with a capacity of up to 350 MW, which will cover the energy consumption of 50,000 households.

In this paper we investigate how to design wind farms, onshore or offshore, so as to maximize the power output. In the focus of our attention is a physical property referred to as the wake effect, i.e., the fact that the wind velocity is reduced downstream behind a turbine and causes a reduced power output from the turbines thus located.

More specifically, we consider the following problems:

Problem 1: We are given a region along with data on the wind conditions, and a number of turbines that are to be placed within the region. We want to find the placement that returns the maximal amount of power. The coordinates of the turbines are to be determined.

Problem 2: Similar to Problem 1, but now the number of turbines is not fixed, and there are only a finite number of possible locations. We take the cost c of installing and running a turbine into consideration, and maximize the net present value of the wind farm. For each possible location, a decision whether or not to install a turbine there is to be made.

To solve these problems, we need a model for the wake effect, and effective optimization methods. In the remainder of the paper, we briefly review the literature supporting these two needs (Section 2), we develop a new wake model (Section 3), we give solution methods for Problems 1-2 (Section 4), and we evaluate the methods experimentally (Section 5).

2 Literature review

Wake models

A method for estimating the wake effect which has become increasingly popular in recent years is to solve some simplified version of the Navier-Stokes equations [3]. Moreover, a commercially available wake effect tool called Fuga [4] solves linearized Reynolds-averaged Navier-Stokes equations for farms with 200 turbines in a matter of seconds.

Over the years, several wake models that are considerably simpler have been proposed. Three well known examples are the Jensen model [5, 8], the Frandsen model [6] and the Larsen model [7].

The wake model used in the current work will be based on the Jensen model as described in [8]. This frequently cited model has been used to this day, or at least until recently, by e.g. the Wind Atlas Analysis and Application Program [9] (‘a PC program for predicting wind climates, wind resources and power productions from wind turbines and wind farms’). We therefore go on to review this model more closely.

The Jensen model

When wind with initial velocity v flows through the rotor of a wind turbine, it will normally lose some kinetic energy to the rotor blades, and the wind velocity immediately

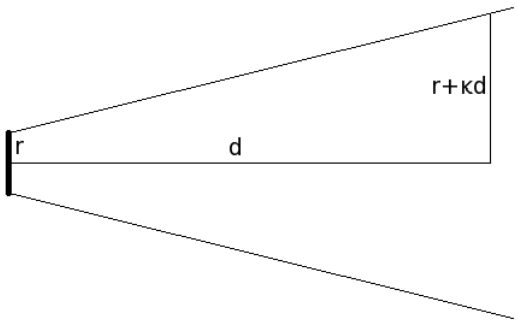

Figure 1: Linear expansion of the wake behind a single turbine

behind the rotor will be αv for some α, $0 < \alpha < 1$. The exact value of α depends on the pitching of the blades. The power coefficient, which denotes the ratio of the power that is extracted from the wind by the turbine to the available power in the wind (assuming an ideal massless rotor and no heat transfer), is given by

$$C_p = \frac{1}{2}(1 + \alpha - \alpha^2 - \alpha^3).$$

A straightforward analysis of this function shows that its maximum on the interval $[0,1]$ is attained for $\alpha = \frac{1}{3}$. This yields Betz' law [10] stating that the maximal value of C_p is $\frac{16}{27} \approx 0.593$. In the Jensen model, it assumed that this maximum is attained.

The kinetic energy of the wind is proportional to v^3. With a fixed value of the power coefficient, it follows that also the power output of a single turbine is a fixed constant times v^3. Thus for simplicity, we will define the *revenue* of a single turbine to be v^3.

Furthermore, the Jensen model is based on a linear expansion of the wake region (Fig. 1), and on mass conservation. This means that the wind velocity in the wake a given distance d behind a turbine can be expressed as $(1-\delta)v$ where δ, called the velocity deficit coefficient, must be $\frac{2}{3}$ for $d = 0$, and is inversely proportional to the area of the corresponding cross section of the wake region. It follows that

$$\delta = \frac{2}{3}\left(\frac{r}{r+\kappa d}\right)^2 \tag{1}$$

up to a distance $r + \kappa d$ from the centre line, where κ is the decay constant.

Katić et al.'s wake combination model

For multiple wakes, the combined deficit would logically appear to be the sum of the individual deficits. However, this estimate is known to compare poorly with observations. In the model of Katić et al. [8], the combined deficit is instead the 2-norm of the vector of individual deficits:

$$v_j = v_0 - \sqrt{\left(\delta_{1j} v_1\right)^2 + \ldots + \left(\delta_{(j-1)j} v_{j-1}\right)^2} \tag{2}$$

where v_0 is the initial wind velocity, v_i is is the wind velocity experienced by turbine i, and δ_{ij} is the deficit coefficient at turbine j caused by turbine i's presence. The authors state that this formula is chosen so that the velocity deficit from a line of turbines will quickly reach an equilibrium level, in agreement with observations.

Wind farm optimization

One of today's perhaps most promising wind farm optimization platforms is TopFarm [11]. Its optimization method is in part a genetic algorithm, which belongs to the class of metaheuristic methods. Although genetic algorithms are considered to be the most popular approach in wind park optimization [12], they can be rather slow [13]. This opens up for the possibility that it is worthwhile to consider simpler methods.

There are a number of heuristic methods in the literature [14]. Trying out all of them is beyond the scope of this paper, and we will instead consider only a few. These will be tested on a set of computer generated cases.

3 An improved wake model

Based on [8], we propose an improved version which is still fairly simple, but appears to fit the measurements better.

The wake behind a single turbine

Our starting point is the wake combination model (2), which in turn is based on the Jensen model (1). The aim of (1) is not to describe the velocity field accurately; a Gaussian distribution of the deficit across the wake is more commonly assumed [8]. A bell-shaped distribution is also evident in Fig. 4 of the article. On this basis, and since a certain level of accuracy in the velocity field description is required for our purpose, we propose a wake model with a Gaussian distribution across the wake, and mass conservation like in the original model. Immeditately behind a turbine, the velocity deficit coefficient is $\frac{2}{3}$ in a region with area πr^2 according to the Jensen model. Therefore, if the velocity deficit coefficient at a distance s from the centre line is given by

$$\delta = \xi e^{-\frac{s^2}{2\sigma^2}}$$

where σ is some parameter corresponding to the standard deviation of a normal distribution, and ξ is some quantity independent of s, then the mass conservation equation becomes

$$\int_0^\infty 2\pi s \xi e^{-\frac{s^2}{2\sigma^2}}\,\mathrm{d}s = \frac{2}{3}\pi r^2 \Rightarrow 2\pi\sigma^2\xi = \frac{2}{3}\pi r^2$$

On solving for ξ, we get $\xi = \frac{r^2}{3\sigma^2}$ and the velocity deficit coefficient is given by

$$\delta = \frac{r^2}{3\sigma^2} e^{-\frac{s^2}{2\sigma^2}}$$

Imposing that δ agrees with (1) for $s = 0$, we get $\frac{r^2}{3\sigma^2} = \frac{2}{3}\left(\frac{r}{r+\kappa d}\right)^2$ yielding $\sigma = \frac{r+\kappa d}{\sqrt{2}}$, which gives

$$\delta = \frac{2}{3}\left(\frac{r}{r+\kappa d}\right)^2 e^{-\frac{s^2}{(r+\kappa d)^2}}. \tag{3}$$

Multiple wakes

Consider the wake combination model (2) for the velocity deficit in the case of several overlapping wakes. In [8], the authors assume $\kappa = 0.075$, but later on they state that in the case of two turbines, it would be better to set κ to 0.11 after the second turbine. This

recommendation is based on measurements done within 8 rotor radii. In [15], it is shown that by using the 3-norm instead of the 2-norm for combined wakes, this inconsistency can be avoided. This approach is also taken in the current work.

In conclusion, if the turbines are sorted in the downstream direction such that $i < j$ means that turbine j is located downstream to turbine i, we apply the wake combination model

$$v_j = v_0 - \sqrt[3]{\sum_{i=1}^{j-1} \delta_{ij}^3 v_i^3}, \tag{4}$$

where δ_{ij} is obtained by inserting $d = d_{ij}$ and $s = s_{ij}$ into (3), and where d_{ij} and s_{ij} are the distances between turbines i and j in, respectively, the wind direction and the direction orthogonal to the wind.

4 Solution methods

Solution algorithms that guarantee the optimal solutions have, in a recent study [15], shown to be too time consuming in instances of realistic size. Consequently, we focus on heuristic methods that are able to produce solutions of high quality quickly, and rely on experimental evaluation of their performance.

Heuristics for Problem 1

To avoid the computational cost of numerical differentiation, we have chosen to apply derivative-free optimization methods. One well known method of this type is the Nelder-Mead method, described by e.g. Nocedal and Wright [16]. The idea of the method is to keep track of $n+1$ points (simplex vertices) in the solution space simultaneously, and at each step, either improve on the worst point along the line through the center of the others, or shrink the whole simplex towards the best point. In our application, each vertex of the simplex contains the coordinates of all the turbines.

Algorithm 1 A heuristic method for Problem 1

$C \leftarrow \{(x_1, y_1), ..., (x_n, y_n)\}$ (random feasible configuration), $e \leftarrow f(C)$
$R \leftarrow R_0$
while $R > R_{\min}$ **do**
  for $i = 1$ **to** N **do**
    for $j = 1$ **to** n **do**
      repeat
        $\Delta x_j \leftarrow$ random value $\in [-R, R]$, $\Delta y_j \leftarrow$ random value $\in [-R, R]$
      until $(x_j + \Delta x_j, y_j + \Delta y_j)$ inside region
    $C' \leftarrow \{(x_1 + \Delta x_1, y_1 + \Delta y_1), ..., (x_n + \Delta x_n, y_n + \Delta y_n)\}$, $e' \leftarrow f(C')$
    if $e' > e$ **then**
      $C \leftarrow C'$, $e \leftarrow e'$
  $R \leftarrow R \times 0.9$
return C; e

The Nelder-Mead method is designed for unconstrained optimization problems, whereas in our problems, the constraints that turbines must be located within a given region are imposed. We propose two different ways to handle the constraints. We can either relax them by introducing a Lagrangian multiplier, or we can perform a truncated

binary search whenever an infeasible vertex is encountered. These two variants will be referred to as Nelder-Mead-a and Nelder-Mead-b. Details can be found in [15].

Algorithm 1 describes our alternative to Nelder-Mead. The following notation is used: A configuration is given by $C = (x_1, y_1), \ldots, (x_n, y_n)$, where (x_i, y_i) denotes the coordinates of turbine $i \in \{1, \ldots, n\}$. The evaluation function is generally denoted by f, such that when the algorithm is applied to Problem 1, $f(C)$ refers to the power generated by choosing configuration C, whereas $f(C)$ means the total revenues when the algorithm is applied to Problem 2. The idea is to try out configurations where each turbine is moved at random from its current position, and to limit the move within a distance R in both the north-south and the east-west directions. We update the best configuration if the new one is found to be better, and repeat the process with a slowly decreasing value of R. For each value of R, a specified number N of trials are made.

Preliminary experiments have shown that it is profitable to locate turbines on the boundary of the feasible region. An undesirable property of our heuristic in the form of Algorithm 1, is that when many turbines are located on the boundary, many of the trial configurations to be tested will either be outside the region (and rejected), or inside the region but off the boundary again. In our implementation, we have therefore modified the algorithm such that turbines placed outside the region are moved to the closest point on the boundary before the new configuration is evaluated.

Algorithm 2 Overview of a heuristic method for Problem 2

$e_{\max} \leftarrow 0$
for $i = 0$ **to** $k-1$ **do**
  for $j = 0$ **to** $k-1$ **do**
    Let the subset of possible turbine locations consist of points covered by grid G_k when its southwestern corner is in position (x_i, y_j)
    Start with no installed turbines
    repeat
      Calculate the net revenues obtained by installing a new turbine for each of the vacant slots
      Install turbine in most profitable vacant slot
    until no improvement possible
    repeat
      Select most profitable action from:
      – installing turbine in vacant slot
      – deinstalling existing turbine
      – moving turbine to nearby vacant slot (for example, within a distance $\sqrt{5}$)
    until no improvement possible
    $e \leftarrow$ net profit
    if $e > e_{\max}$ **then**
      $e_{\max} \leftarrow e$, Store configuration
return best configuration found, $e_{\max}$

Heuristics for Problem 2

For Problem 2, we suggest the heuristic method given in Algorithm 2. Assume that the set of possible turbine locations are defined by a rectangular grid $\{(x_i, y_j) : i, j = 0, \ldots, m\}$, where $x_1 - x_0 = \cdots = x_m - x_{m-1} > 0$ and $y_1 - y_0 = \cdots = y_m - y_{m-1} > 0$. The

heuristic generates different subsets of this grid, and each point in the subsets is referred to as a *slot*. For a given *mask width* k (assume for simplicity that k divides m), the first subset is $\{(x_i, y_j) : i, j = 0, k, 2k, \ldots, m\}$, the next subset is $\{(x_i, y_j) : i = 0, k, 2k, \ldots, m, j = 1, k+1, 2k+1, \ldots, m-k+1\}$, and so on up to the last subset $\{(x_i, y_j) : i, j = k-1, 2k-1, \ldots, m-1\}$. In other words, the heuristic considers each possible subset that can be covered by putting a coarser grid, G_k, where every kth grid line in each direction is included, on top of the original grid.

Iit is not always practically feasible to install a turbine at any point on a rectangular grid. Our heuristic is therefore designed such that it accepts as input a binary matrix describing at what grid points installation is allowed. In Fig. 2, we show an example where only the central part (shaded) of the rectangular area can receive a turbine. The figure also shows the slots considered by Algorithm 2 in the case of $k = 3$ when $i = 2$ and $j = 1$.

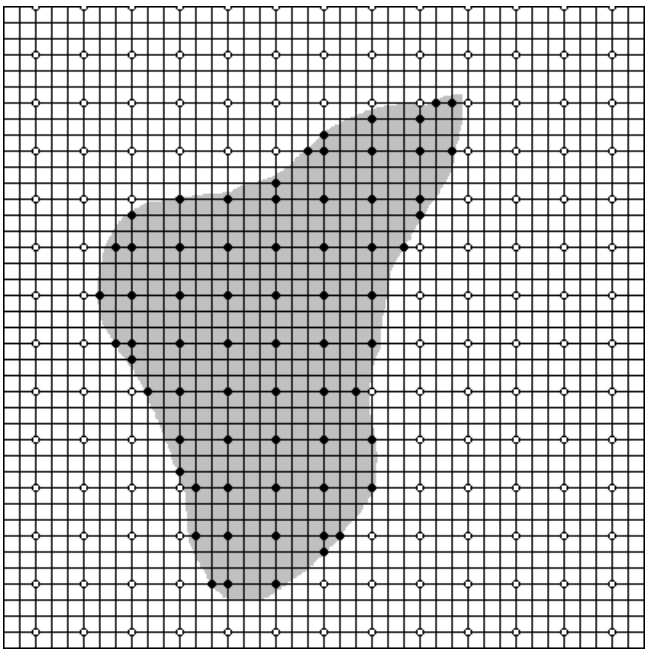

Figure 2: (Feasible) grid points and associated slots for $k = 3$, $i = 2$ and $j = 1$

Since boundary points are assumed to be particularly attractive [15], we also implement a variant of Algorithm 2 where also extreme feasible grid points are included as slots. By this we mean grid points that are either the leftmost or the righmost feasible grid point on its horizontal grid line. A similar definition applies to extreme feasible grid points in the vertical direction (see Fig. 2).

5 Experiments

We report results from three sets of experiments: For Problem 1, we compare the performance of Algorithm 1 to the two variants of the Nelder-Mead algorithm. For Problem 2, we run Algorithm 2, with and without boundary points included, for different mask widths, and compare running time and quality of the output. Finally, we make a comparison of Algorithms 1 and 2 when applied to Problem 2. To this end, the former, which in the form given in Section 4 assumes a continuous set of turbine locations, must be adapted to work on a discrete set of points. This is however straightforward, since the only modification needed is to draw the random values of Δx_j and Δy_j from a discrete

probability distribution, and to ensure that no two identical coordinate pairs are drawn. We will refer to this algorithm as Algorithm 1-disc.

Test instances

One way of representing the area occupied by a wind farm is to provide a set of linear inequalities. This is suitable for the heuristic methods we considered for Problem 1. Both variants of the Nelder-Mead algorithm, as well as Algorithm 1, accept input on this form. For the experiments with these algorithms, we have generated 25 cases, labelled lc01, ..., lc25, randomly based on linear inequalities. In each case, we have $n = 20$ turbines with rotor radius $r = 1.5$.

For Problem 2, we have generated 50 instances, labeled bm01, ..., bm50, represented by a binary matrix, in which each entry corresponds to a pixel in the farm that may or may not receive a turbine. In each instance, we let $m = 99$, which means that the grid of potential locations contains $(m+1)^2 = 10000$ points. The rotor radius is set to $r = 1.0$, and the turbine cost is set to $c = 60$.

In all instances of both problems, we have included the 32 wind directions defining the angles $0, \frac{\pi}{16}, \ldots, \frac{31\pi}{16}$ with the west-east direction. Probabilities and velocities are generated for each wind direction such that the differences in their values for two neighboring directions is relatively small. More details about the instance generation are given in [15], and the cases can be found on the website [17].

Experimental results

In order to tune some of the parameters, we first compute the expected revenues, Ω, of installing a single turbine on the farm.

Experiments with methods for Problem 1

For the Nelder-Mead algorithms, the stopping criterion is that the difference between the evaluations of the best and the worst vertex becomes smaller than a parameter $\Delta = 10^{-7}\Omega$, and the Lagrangian coefficient in Nelder-Mead-a is set to 0.8Ω. For Algorithm 1, the initial value of R, R_0, is set to 50.0, and $R_{\min}$ is set to 10^{-5}. We choose the number of trials, N, such that the total running time becomes comparable to the running time of the Nelder-Mead variants, and preliminary experiments hence suggested $N = 100$. We record the best of 10 runs in each instance.

The results of the experiments are given in Table 1. An upper bound on the total revenues can be found by disregarding the wake effect, and is given by $n\Omega$. The columns with heading 'Rel. revenues' contain the revenues relative to this upper bound, and the columns with heading 'Time' contain the mean CPU running time for the 10 runs.

We observe that the two Nelder-Mead algorithms performed almost equally well, while our Algorithm 1 consistently produced better solutions at a modest increase in computational cost. Algorithm Nelder-Mead-a occasionally produced infeasible solutions. Even though the violations of the linear inequalities were mostly very small, they were so rare that they were simply discarded.

From the last line of Table 1 we can see that the mean running time for Nelder-Mead-b was about $\frac{1}{6}$ longer than that of Nelder-Mead-a, while the mean running time for Algorithm 1 was just slightly longer than that of Algorithm Nelder-Mead-b (by about 6%). For a given algorithm, we would expect the mean running times for the individual cases to be influenced by the number of linear inequalities, but initial configurations can

Table 1: Best relative revenues in 10 runs

	Nelder-Mead-a		Nelder-Mead-b		Algorithm 1	
Case	Rel. revenues	Time	Rel. revenues	Time	Rel. revenues	Time
lc01	0.9410	63.7s	0.9442	65.5s	0.9534	75.4s
lc02	0.9392	60.1s	0.9369	68.9s	0.9494	73.7s
lc03	0.9332	53.8s	0.9362	88.4s	0.9447	74.2s
lc04	0.9275	51.6s	0.9188	73.0s	0.9379	76.7s
lc05	0.9522	59.7s	0.9530	66.3s	0.9591	71.8s
lc06	0.9488	57.9s	0.9505	76.0s	0.9578	72.7s
lc07	0.9541	72.9s	0.9545	81.2s	0.9640	77.7s
lc08	0.9378	48.1s	0.9360	48.6s	0.9477	51.5s
lc09	0.9400	50.5s	0.9414	60.8s	0.9515	66.3s
lc10	0.9120	59.0s	0.9165	72.9s	0.9333	76.3s
lc11	0.9206	43.4s	0.9210	45.2s	0.9360	57.3s
lc12	0.9001	71.3s	0.8995	61.9s	0.9165	73.7s
lc13	0.9172	42.2s	0.9116	48.8s	0.9291	58.3s
lc14	0.9130	60.2s	0.9098	77.4s	0.9277	78.2s
lc15	0.9242	51.1s	0.9236	44.0s	0.9383	62.8s
lc16	0.9265	59.8s	0.9255	80.5s	0.9361	78.5s
lc17	0.9515	66.1s	0.9467	75.9s	0.9586	80.8s
lc18	0.9532	47.4s	0.9555	47.5s	0.9683	56.2s
lc19	0.9449	54.4s	0.9497	59.3s	0.9640	67.6s
lc20	0.9414	59.5s	0.9392	75.7s	0.9490	78.0s
lc21	0.8627	52.3s	0.8767	63.0s	0.8955	64.7s
lc22	0.9210	77.4s	0.9262	71.3s	0.9489	80.0s
lc23	0.9432	49.6s	0.9455	59.1s	0.9538	61.4s
lc24	0.9301	64.8s	0.9320	64.9s	0.9408	75.6s
lc25	0.9222	52.2s	0.9217	64.3s	0.9335	73.2s
Mean	0.9303	57.2s	0.9309	65.6s	0.9438	70.5s

probably also have an impact. For example, for Algorithm 1, the case with the longest running time was wf37, which has only four linear inequalities.

Experiments with methods for Problem 2

For Problem 2, Algorithm 2 with mask widths 7, 5 and 3 was applied to the cases based on binary matrices. As discussed in Section 4, we applied both versions of the algorithm, with boundery points excluded and boundery points included, respectively. In some of the cases, the algorithm suggested that no turbine be installed, due to the fact that $\Omega \leq c$. The results from the remaining cases are given in Table 2. The first line of each entry contains the best net profit and the corresponding number of turbines to be installed, and the second line contains the running time.

Table 2 shows the expected results that a smaller mask width usually (not always) gives better results for Algorithm 2, but generally leads to a longer running time. Due to the smaller number of possible slots to be evaluated, the algorithm is faster when boundary points are excluded. However, including the boundary points gives consistently better results than excluding them, even if a coarser grid is applied. Using mask width

Table 2: Experimental results of Algorithm 2

	Boundary points excluded			Boundary points included		
Case	Mask w. 7	Mask w. 5	Mask w. 3	Mask w. 7	Mask w. 5	Mask w. 3
bm02	279.6 (27)	292.0 (28)	315.6 (30)	328.6 (32)	338.1 (34)	352.3 (35)
	1m53s	3m1s	15m18s	7m48s	10m5s	34m55s
bm08	98.14 (17)	101.9 (17)	103.2 (18)	105.8 (18)	107.1 (19)	107.9 (19)
	51s	1m33s	9m24s	2m37s	5m9s	14m4s
bm09	1044 (53)	1099 (58)	1173 (62)	1214 (66)	1242 (64)	1304 (69)
	7m7s	18m14s	66m1s	36m9s	50m51s	135m34s
bm10	31.82 (9)	32.92 (10)	33.37 (9)	34.98 (11)	35.59 (11)	34.63 (10)
	12s	22s	2m1s	37s	52s	4m1s
bm13	435.6 (33)	471.3 (35)	502.5 (40)	514.0 (37)	539.0 (41)	549.8 (41)
	2m54s	5m42s	23m28s	10m29s	18m22s	48m9s
bm18	64.12 (14)	65.88 (13)	68.36 (15)	69.00 (15)	70.24 (15)	70.24 (15)
	36s	1m1s	6m23s	1m42s	3m12s	13m16s
bm20	5000 (86)	6224 (164)	6868 (197)	6208 (119)	7191 (203)	7523 (217)
	17m52s	124m44s	1025m42s	83m48s	399m16s	1889m57s
bm26	0.1880 (2)	0.1760 (2)	0.1780 (2)	0.1926 (2)	0.1958 (2)	0.1846 (2)
	3s	5s	17s	8s	12s	35s
bm29	3694 (72)	4446 (130)	4874 (151)	4624 (103)	5079 (165)	5293 (167)
	8m16s	58m13s	424m7s	49m42s	230m12s	763m37s
bm43	48.52 (11)	48.90 (11)	51.10 (12)	53.14 (12)	53.57 (12)	54.97 (13)
	14s	23s	1m45s	46s	1m6s	4m11s
bm44	623.7 (41)	650.2 (41)	693.0 (44)	713.0 (46)	739.5 (47)	762.1 (49)
	4m4s	7m10s	28m9s	15m3s	23m35s	62m14s
bm45	13.41 (7)	14.04 (8)	14.05 (7)	14.36 (8)	14.43 (8)	14.75 (8)
	14s	25s	2m30s	36s	1m7s	4m25s
bm47	1863 (62)	2063 (76)	2371 (92)	2298 (84)	2396 (89)	2588 (99)
	5m12s	27m28s	113m59s	33m58s	81m30s	221m51s
bm49	294.1 (30)	294.5 (31)	315.6 (32)	324.5 (35)	330.9 (35)	333.5 (34)
	2m55s	4m59s	27m18s	9m53s	17m26s	47m59s
bm50	10.53 (6)	10.64 (7)	10.76 (6)	11.75 (7)	11.84 (7)	11.37 (7)
	11s	21s	2m0s	30s	46s	3m53s

5 and including the boundary points gives, for example, better results than mask width 3 in combination with exclusion of the boundary points. Since a comparison between the second row entries in columns 4 and 6 in Table 3 shows that the former approach also is the faster, our recommendation is to apply Algorithm 2 with inclusion of boundary points.

Experiments with comparisons of Algorithms 1-disc and 2

For the cases in which $\Omega > c$, we ran Algorithm 1-disc with n determined by the output from Algorithm 2. The search radius R starts at $R_0 = 50$ and terminates at $R_{\min} = 1$. This yields fewer different values of R than in the experiments with Problem 1, and therefore, N has been increased to 1000.

The results of the last experiments are given in Table 3. The second column ('n') contains the number of turbines, based on the results given in Table 2. The third column ('Revenues') contains the best revenues found in 10 runs of Algorithm 1-disc. The next

column ('Net profit') contains the differences between the entries in the third column and the costs nc with the values for n given in the second column (recall that $c = 60.0$). These values can then be compared with the entries in Table 2. The last column contains the total running times for the 10 runs.

From Tables 2-3 we can see that Algorithm 2 tends to give better results than Algorithm 1-disc, except in extreme cases with very few turbines, such as bm26 and bm50. With the same exceptions, the running time of Algorithm 1-disc is of the same order of magnitude as that of Algorithm 2 (boundary points included) with mask width 3. However, when the (optimal) number of turbines is large, Algorithm 2 with larger mask width still gives better results than Algorithm 1-disc, while being faster.

The large variations in the running times seen in Table 3 can be explained by the recursive nature of (4). This implies that the number of calculations that must be carried out in each step of Algorithm 1-disc is $O(n^2)$.

Table 3: Experimental results of Algorithm 1-disc

Case	n	Revenues	Net profit	Running time
b02	35	2381.0	281.0	57m32s
b08	19	1246.2	106.2	17m1s
b09	69	5172	1032	213m5s
b10	11	695.54	35.54	5m12s
b13	41	2936.5	476.5	70m36s
b18	15	968.88	68.88	10m49s
b20	217	18913	5893	1959m25s
b26	2	120.1963	0.1963	13s
b29	167	14206	4186	1092m20s
b43	13	834.23	54.23	7m19s
b44	49	3570.6	630.6	109m20s
b45	8	495.06	15.06	3m3s
b47	99	7940	2000	386m58s
b49	36	2461.2	301.2	63m35s
b50	7	431.78	11.78	2m19s

6 Conclusion

We have suggested computational methods for maximizing the total power and the net profit that can be generated from a wind farm. To reflect the interference between the turbines in the farm, we have developed an improved version of a popular wake model known as the Jensen model. We have also adopted a recently suggested improvement of a popular wake combination model for handling interference from multiple turbines.

Exact optimization methods are prohibited by the difficulty of the problem under study and the size of realistic instances, and therefore we have suggested fast heuristics that do not guarantee to find optimal solutions. For the problem of maximizing power generation from a given number of turbines, our method operates on continuous variables for turbine locations. When optimizing also the number of turbines to be installed such that the total net profit is maximized, we have used a discrete representation of possible locations. Our experiments document that the proposed methods give good results when applied to

instances of realistic size, and show good flexibility in balancing fast computation against quality of the output.

References

[1] http://en.wikipedia.org/wiki/Wind_farm

[2] http://vestavindoffshore.no/havsul

[3] Y. Heggelund, I.M. Skaar: Fast evaluation of wind farm flow for use in layout optimization, NORCOWE-RR-C-10-WP4-001.

[4] S. Ott, J. Berg and M. Nielsen: Linearised CFD Models for Wakes, Report Risø-R-1772(EN), Risø National Laboratory, Roskilde, Denmark, December 2011, ISBN 978-87-550-3892-9.

[5] N.O. Jensen: A note on wind generator interaction, Report Risø-M-2411. Risø National Laboratory, Roskilde, Denmark, 1983.

[6] S. Frandsen, R. Barthelmie, S. Pryor, O. Rathmann, S. Larsen, J. Højstrup and M. Thøgersen: Analytical modelling of wind speed deficit in large offshore wind farms, *Wind Energy* 9(1-2): 39-53 (2006).

[7] G.C. Larsen: A Simple Wake Calculation Procedure, Risø M-2760, Risø National Laboratory, Roskilde (Denmark), 1988.

[8] I. Katić, J. Højstrup, N.O. Jensen: A simple model for cluster efficiency, *Proceedings of EWEC'86*, Rome, Italy (1986), 407–410.

[9] http://www.wasp.dk/Products/WAsP/WakeEffectModel.html

[10] http://en.wikipedia.org/wiki/Betz'_law

[11] P.-E. Réthoré, P. Fuglsang, G.C. Larsen, T. Buhl, T.J. Larsen, H.A. Madsen: TopFarm: Multi-fidelity Optimization of Offshore Wind Farm, *Proceedings of the Twenty-first (2011) International and Polar Engineering Conference*, 516–524.

[12] http://windenergyresearch.org/2010/10/state-of-the-art-in-wind-farm-layout-optimization

[13] http://en.wikipedia.org/wiki/Genetic_algorithms

[14] http://en.wikipedia.org/wiki/Mathematical_optimization\#Heuristics

[15] J.K. Haugland: Optimization Models for Turbine Location in Wind Farms, M.Sc. Thesis, Department of Informatics, University of Bergen, May 24, 2012, http://www.neutreeko.net/thesis1.pdf.

[16] J. Nocedal, S.J. Wright: *Numerical Optimization,* Springer, 2006 (second edition), 238–240.

[17] http://www.neutreeko.net/wind.htm