An Adaptive Model-based Mutation Operator for the Wind Farm Layout Optimisation Problem

Michael Mayo, Maisa Daoud
Department of Computer Science
University of Waikato
Hamilton, New Zealand
mmayo@waikato.ac.nz, maysa_taheir@yahoo.com

***Abstract*—A novel mutation operator for the wind farm layout optimisation problem is proposed and tested. When a wind farm layout is simulated, statistics such as an individual turbine's wake free ratio can be computed. These statistics are in addition to the global measure being optimised, for example the overall cost of energy extraction of the farm. We present algorithms that first of all build a predictive model of the wake free ratio across an entire wind farm. This model is then used inside a mutation operator to perturb turbines towards positions of high predicted wake free ratio. We evaluate our approach by comparing a 1+1 Evolutionary Strategy using this new mutation operator vs. the same algorithm with a more standard random mutation operator, and show that our new operator leads to the discovery of wind farm layouts having a statistically significantly lower cost of energy extraction.**

***Keywords*-wind farm layout optimisation problem, evolutionary strategy, mutation operator, wake free ratio, cost of energy, machine learning, predictive model, stochastic hill climbing**

I. Introduction

An important issue in wind farm design is assigning location and other properties to the turbines on the farm in order to optimise some metric such as the cost of wind energy extraction. This problem is by no means trivial: to illustrate, if each turbine is specified solely by an (x, y) position (meaning that all the turbines are identical and have no other properties), then for a 400-turbine layout, the problem amounts to an 800-dimensional optimisation task. Solving such a problem analytically is likely to be either extremely difficult or impossible. To further complicate the problem, many additional factors must also be taken into account: for example the farm's initial construction and maintenance costs is likely to be significant and so should be included in any cost estimates; and the aesthetic impact of the farm on the environment should also be considered.

Given so many conflicting objectives, meta-heuristic search algorithms are an ideal tool to find solutions.

The primary novel contribution of this paper is an adaptive model-based mutation operator for wind farm layouts that can be used to solve the wind farm layout optimisation problem. We focus on the more difficult *continuous* version of the layout optimisation problem in which turbines may be placed at any valid location on the farm. In comparison, other earlier approaches to solving this problem often constrained turbine placement, for example to discrete locations on a grid (an approached utilised by seminal works in this area, such as Mosetti et al. [4]). Our approach also copes with obstacles of any shape and size.

The metaheuristic search algorithm that we use in this paper is the Evolutionary Strategy (ES, [1]), which in its simplest form (the 1+1 variant) is equivalent to the well-known stochastic hill climbing algorithm.

Traditionally, search algorithms such as ESes rely on the availability of a value function in order to provide a single numeric measure of the value of a particular solution. It is this single measure that the ES optimises. Wind farm simulation models, however, provide a much richer description of the wind farm's performance than a single objective value. Each turbine can have a multitude of statistics computed about it as a wind farm simulation executes. These are basically "side effects" of the simulation. An example used in this research the *wake-free ratio*, a metric representing the theoretical maximum wind energy that a turbine can receive which is *not* degraded by the wakes of other nearby turbines.

In this paper, we show that a predictive model of the wake free ratio across a wind farm layout can be built based on two factors: the absolute position of the turbine, and the local configurations of nearby neighbouring turbines. This model can then be used to predict wake free ratios at vacant positions in the layout, which in turn can be used to improve the performance of the ES algorithm's mutation operator.

We evaluate our model-based mutation operator against a random mutation operator on the twenty benchmark scenarios proposed for the 2015 Wind Farm Layout Optimisation competition [7] and obtain statistically significant improvements in the cost of energy extraction across all scenarios.

II. Background

A. Wind Farm Layout Optimisation Problem

The Wind Farm Layout Optimisation Problem concerns finding an optimal placement for wind turbines (and optionally setting their properties as well) on a wind farm [5]. Typically, the number of wind turbines that must be positioned is high (for example, in the hundreds of turbines in more recent publications) and there may be obstacles (e.g. water) on the

978-1-4799-8697-2/15/$31.00

farm where turbines cannot be placed. Furthermore, turbines cannot be placed too closely to each other because of the extreme turbulence that they generate. According to Samorani [5], a minimum distance three times the diameter of the wind turbine rotors is sufficient to avoid damage.

Despite this minimum distance constraint, nearby wind turbines still interact. The primary mechanism for this is known as the *wake effect*. Essentially, the wake effect is a "spreading cone" of slow, turbulent air downwind of a turbine that reduces the amount of harvestable energy by other turbines in its path. Wake effects from multiple turbines amalgamate if there are many nearby turbines upwind.

Evaluating the performance of a particular wind farm layout is quite challenging therefore: factors for consideration include the distribution of wind speeds and directions (which are often characterised using a *wind rose*, see Figure 2 for examples); the wake effects between turbines; as well as the particular characteristics of the turbines themselves (for example, different turbines may have different *cut out* speeds at which they switch off if the wind speed is too strong). A complex simulation is therefore required in order to estimate the total energy output of a farm. With respect to wake effect modelling, approaches range from extremely complex fluid dynamics simulations [5] to relatively less complex models such as the Park model (which none-the-less has time-complexity $\mathcal{O}(n^2)$ in the number of turbines [6]).

B. 1+1 Evolutionary Strategy

In this paper we utilise one of the simplest metaheuristic optimisation strategies, namely the 1+1 ES [1]. Pseudocode for the algorithm in the context of a cost minimisation problem is depicted as Algorithm 1.

Input: MAX_EVALS
begin
  $best \leftarrow create\_random\_solution()$;
  $best\_cost \leftarrow evaluate(best)$;
  $num\_evals \leftarrow 1$;
  repeat
    $candidate \leftarrow mutate(copy(best))$;
    $candidate\_cost \leftarrow evaluate(candidate)$;
    $num\_evals \leftarrow num\_evals + 1$;
    if $candidate\_cost \leq best\_cost$ **then**
      $best \leftarrow candidate$;
      $best\_cost \leftarrow candidate\_cost$;
    end
  until $num\_evals \geq MAX\_EVALS$;
  return $best$
end

Algorithm 1: 1+1 Evolutionary Strategy.

There are two primary reasons for adopting the 1+1 ES in this research. Firstly, the algorithm is simple to implement and understand, and it requires only a minimum number of parameters (specifically, the maximum allowable number of solution evaluations, which is a critical parameter in the wind farm layout optimisation problem because evaluation is expensive). This reduced number of parameters makes the 1+1 ES highly amenable to experimentation as the effects of different parameter settings on experimental results can be minimised.

The second reason for utilising this algorithm is that it requires a problem-specific mutation operator only. Other more sophisticated algorithms such as the genetic algorithm require more operators to be specified, for example both a mutation and a crossover operator. Since this research is focused on a problem-specific mutation operator for the wind farm layout optimisation operator, using the 1+1 ES makes sense. (In fact, in our initial experiments with a genetic algorithm, we found that a poor choice of crossover operator significantly degrades the algorithm's overall performance.)

III. Adaptive Model-Based Mutation of Wind Farm Layouts

Both the mutation operator and the objective function in a 1+1 ES are problem-specific. In this section, we discuss our approach to designing a mutation operator that is specific to the wind farm layout optimisation problem.

To begin with, Algorithm 2 is pseudocode for a simple randomised mutation operator for wind farm layouts. It moves a small number of turbines (for example, 5%) to a new random location. This algorithm represents the simplest possible approach to mutating a wind farm layout. The mutation rate parameter, an input to the algorithm, dictates the size of the random layout changes – this is compared to a uniform random number between 0 and 1 to determine which turbines are to be moved. Such an approach is typical in generic evolutionary strategies and makes no use of problem specific information except for the $random\_valid\_location()$ function which selects a random point on the layout subject to the constraints (i.e. a point is invalid if it lies on an obstacle or is too close to an existing turbine).

Input: turbine positions $T = \{(x_1, y_1), (x_2, y_2), ...\}$,
  mutation rate MUT_RATE
repeat
  for *each turbine position* $(x_i, y_i) \in T$ **do**
    if $random(0, 1) < MUT\_RATE$ **then**
      $(x_{new}, y_{new}) \leftarrow random\_valid\_location()$;
      replace (x_i, y_i) with (x_{new}, y_{new}) in T;
    end
  end
until *the layout has been changed*;

Algorithm 2: Baseline mutation operator.

Wind farm layouts must be simulated in order to evaluate their effectiveness, however, and there is more information available than simply the objective value of a layout and whether or not a vacant point is valid. In fact, the simulation we are utilising calculates the wake free ratios for each wind turbine in the layout, and the approach proposed in this paper exploits that information.

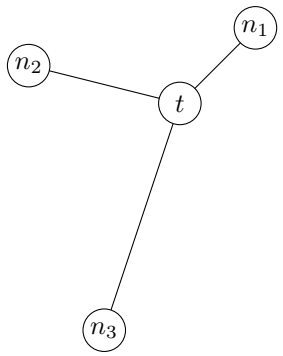


Fig. 1: Example of turbine t's neighbourhood where $K = 3$. The three closest turbines to t are n_1, n_2 and n_3.

To explain, we first of all introduce the concept of a *local neighbourhood* of wind turbines. A local neighbourhood is defined by a neighbourhood size parameter K, and it simply consists of

- the (x, y) position of the turbine on the layout, and
- the K nearest neighbours to the turbine.

An example of a local neighbourhood is given in Figure 1. The motivation for this concept is that, intuitively, a wind turbine's particular performance (i.e. its wake free ratio) is most likely to be influenced by its absolute position on the farm as well as the relative positions of its nearest neighbours.

For example, if a turbine is being placed at the edge of the layout that is nearly perpendicular to the strongest wind direction, this may be advantageous for the turbine. This information is best captured by the turbine's *absolute* position. However, if that location also contains several other turbines partially upwind of the position being considered, the position may actually turn out to be disadvantageous for the turbine because of local wakes. That information is best captured by the *relative* position of the turbine with its neighbours.

The notion of a local neighbourhood of turbines is also used by the Turbine Displacement Algorithm [6], but in that algorithm the local neighbourhood's only purpose is to calculate a turbine displacement vector in a relatively simple way, and absolute position is ignored. Here, we use a more sophisticated approach that exploits both absolute and relative information, and uses that information to build a predictive model.

In order to represent a local neighbourhood, we record the (x, y) position of the central turbine under consideration along with the relative polar distance from (x, y) to each of the turbine's closest neighbours. In the figure, therefore, the small local neighbourhood of size $K = 3$ would be represented by one 2D coordinate and three relative polar coordinates for n_1, n_2 and n_3. The dimensionality required to specify the neighbourhood is therefore eight in the example, or $2(K + 1)$ in general.

We use the absolute position and the local neighbourhood of turbines in the layout to construct a predictive model that can evaluate vacant positions in the layout. Algorithm 3 lists the exact steps required to achieve this.

Input: turbine positions $T = \{(x_1, y_1), (x_2, y_2), \dots\}$,
wake free ratios $W = \{w_1, w_2, \dots\}$,
neighbourhood size K
begin
  $D \leftarrow create\_empty\_dataset()$;
  for *each turbine position* $(x_i, y_i) \in T$ **do**
    $knn \leftarrow k\_nearest\_neighbours(K, (x_i, y_i))$;
    for *each neighbour* $(x_j, y_j) \in knn$ **do**
      $d_j \leftarrow distance((x_i, y_i), (x_j, y_j))$;
      $\theta_j \leftarrow angle((x_i, y_i), (x_j, y_j))$;
    end
    $sort\_by\_distance(knn)$;
    $ex \leftarrow example(x_i, y_i, d_1, \theta_1, \dots, d_K, \theta_K, w_i)$;
    $add\_example(D, ex)$;
  end
  $P \leftarrow build\_model(D)$;
  return P
end

Algorithm 3: Model building algorithm. It is assumed that each turbine has an associated wake free ratio, i.e. $|T| = |W|$.

In more detail, the algorithm constructs a dataset in which examples (i.e. rows) are $2(K + 1)$-dimension specifications of local neighbourhoods. Examples consist of the (x, y) position of the turbine plus the relative polar positions of the K nearest neighbours, in ascending order of distance. Once the example is constructed, it is labelled with the wake free ratio of the turbine at (x, y), a metric which is available after the wind farm layout has been simulated. The example is then added to the dataset, and this is repeated for each turbine in the layout.

Once the dataset is complete, a predictive model is learned from the data using any standard machine learning for regression algorithm. The model is returned by the algorithm.

We now define in Algorithm 4 a more advanced mutation operator than that shown in Algorithm 2. This new mutation operator utilises the model produced by Algorithm 3 to enhance the effectiveness of the 1+1 ES. Briefly, for each turbine that is selected for displacement, N random valid points on the layout are chosen. The predictive model P is then used to predict the wake free ratio at each of the N points, and the point with the best prediction is chosen as the turbine's new location.

One issue concerning our approach is the computational overhead of repeatedly learning the predictive model. There are two issues here: the frequency with which Algorithm 3 is invoked as the ES runs, and the computational overheads of the model. With regards to the first issue, we chose a compromise between building the model once and once only vs. running it after every single change to the best layout. This compromise is to re-learn the model (i.e. invoke Algorithm 3) at regular intervals of every 100 evaluations. The second issue is the computational overhead of the model itself. However, since different machine learning algorithms have vastly different learning and prediction algorithm complexities, overall complexity depends on the choices made which is not this study's

Input: turbine positions $T = \{(x_1, y_1), (x_2, y_2), ...\}$, mutation rate MUT_RATE, number of attempts N, predictive model P

repeat
  for *each turbine position* $(x_i, y_i) \in T$ **do**
    if $random(0, 1) < MUT\_RATE$ **then**
      $(x_{best}, y_{best}) \leftarrow random\_valid\_location()$;
      $w_{best} \leftarrow predict\_wfr(P, (x_{best}, y_{best}))$;
      if $N > 1$ **then**
        for $j = 2 \dots N$ **do**
          $(x, y) \leftarrow random\_valid\_location()$;
          $w \leftarrow predict\_wfr(P, (x, y))$;
          if $w > w_{best}$ **then**
            $(x_{best}, y_{best}) \leftarrow (x, y)$;
            $w_{best} \leftarrow w$;
          end
        end
      end
      replace (x_i, y_i) with (x_{best}, y_{best}) in T;
    end
  end
until *the layout has been changed*;

Algorithm 4: Adaptive model-based mutation operator.

focus. In practice however we found no adverse computational overheads because the datasets used are typically small in machine learning terms.

IV. Evaluation

We use the 2015 Wind Farm Layout Optimisation competition sample scenarios [7] for evaluation. The metric to optimise is the cost per kilowatt produced by the wind farm, subject to minimum proximity constraints between turbines and the presence of obstacles where turbines cannot be placed. To calculate the cost per kilowatt, the evaluation function first of all computes the total power output of the farm after accounting for wake effects as described in [8] and [3]. Once the farm's power output is known, this can be combined with an estimate of the farm's total cost (incorporating multiple factors such as construction cost, yearly operating costs, and interest) to arrive at the average cost per kilowatt [3].

There are twenty scenarios in this competition dataset, and the scenarios are paired, i.e. there are ten different wind direction/speed profiles, and they either do not contain obstacles (scenarios 00...09) or they do contain obstacles (scenarios obs_00...obs_09) .

The wind speed profiles for the scenarios are depicted in Figure 2. Figure 4 depicts one of the layouts with obstacles.

We apply two versions of an ES to each of the twenty scenarios. The first version of the ES uses the simple random layout mutation operator (Algorithm 2), while the second version uses our model-based mutation operator (Algorithm 4). Since ES is a randomised search strategy, the algorithms are executed 10 times on each scenario so that the average best cost per kilowatt across runs can be computed.

TABLE I: Results with a mutation rate of 1%.

Scenario	Baseline	Model-based	p value
00	0.0013583901	0.0013542407	2.6×10^{-7}
01	0.0007969357	0.0007958982	2.6×10^{-5}
02	0.0017544719	0.0017470270	5.1×10^{-10}
03	0.0014432728	0.0014368881	2.2×10^{-11}
04	0.0015679967	0.0015607555	4.4×10^{-12}
05	0.0011646289	0.0011605462	2.3×10^{-9}
06	0.0010474572	0.0010454045	9.2×10^{-7}
07	0.0011448322	0.0011420824	1.4×10^{-5}
08	0.0010458873	0.0010433371	1.4×10^{-7}
09	0.0010152074	0.0010121736	2.3×10^{-13}
obs_00	0.0013584928	0.0013552892	1.7×10^{-8}
obs_01	0.0007974060	0.0007963625	2.5×10^{-4}
obs_02	0.0017552517	0.0017477544	4.6×10^{-12}
obs_03	0.0014444463	0.0014393110	1.1×10^{-8}
obs_04	0.0015684290	0.0015615758	1.4×10^{-11}
obs_05	0.0011655843	0.0011611535	4.4×10^{-11}
obs_06	0.0010479667	0.0010457651	1.2×10^{-10}
obs_07	0.0011450053	0.0011425157	1.9×10^{-6}
obs_08	0.0010459960	0.0010435213	3.7×10^{-8}
obs_09	0.0010158474	0.0010126265	5.0×10^{-12}
all	0.001234175	0.001230211	

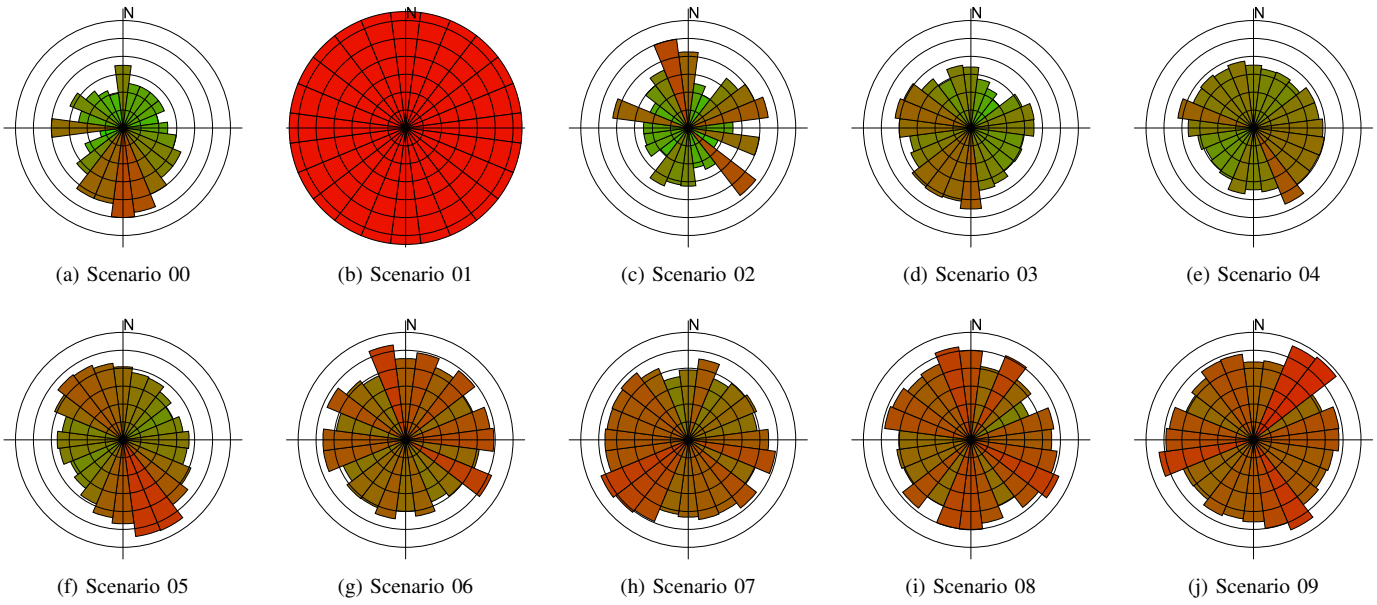
In all runs, the number of turbines is fixed at 400 and the maximum number of evaluations (i.e. the MAX_EVALS parameter) per scenario is 1000. The N parameter in Algorithm 4 is set to 10 and the neighbourhood size parameter K is also fixed at 10. The specific predictive model that we use is an implementation of the Random Forest algorithm adapted for regression [2] with 100 trees.

The mutation rate is an important factor influencing performance in ESes, and to this end we test three different mutation rates: 1%, 5% and 15%, representing small, medium, and large amounts of mutation respectively.

The results are depicted in Tables I, II and III. Each table contains the average cost of energy extraction by both scenario and algorithm, as well as an overall average comparing both algorithms. Since there are ten runs per scenario/algorithm combination, we also performed a statistical test to determine the probability that the mean performance of both algorithms is the same. The p-values resulting from these tests are given. P-values less than 0.01 indicate a greater than 99% probability that the means are different.

The first noteworthy observation to make is that across all algorithms, a lower mutation rate is more effective. For example, the 1+1 ES with a model-based mutation operator achieves best mean costs of 0.001230211, 0.001234030, and 0.001236389 respectively for mutation rates of 1%, 5% and 15%. A similar pattern is followed the 1+1 ES algorithm with the random mutation operator.

The second noteworthy finding is that the ES with the adaptive model-based mutation operator consistently outperforms the baseline algorithm on all scenarios regardless of



(a) Scenario 00 (b) Scenario 01 (c) Scenario 02 (d) Scenario 03 (e) Scenario 04

(f) Scenario 05 (g) Scenario 06 (h) Scenario 07 (i) Scenario 08 (j) Scenario 09

Fig. 2: Wind profiles used in each scenario. Depicted are the wind roses, which give the average wind speed in each direction. Directions are discretised into 15° bins. Each concentric circle in a rose represents a wind speed increase of 2 m/s. Wind speeds are usually less than 12 m/s except in Scenario 01, which has a uniform average wind speed of 13 m/s in all directions.

TABLE II: Results with a mutation rate of 5%.

Scenario	Baseline	Model-based	p value
00	0.0013642280	0.0013592191	1.3×10^{-10}
01	0.0008003460	0.0007984432	2.6×10^{-9}
02	0.0017626870	0.0017528339	2.4×10^{-12}
03	0.0014505825	0.0014424547	4.6×10^{-12}
04	0.0015757560	0.0015654734	2.2×10^{-13}
05	0.0011695491	0.0011637115	3.8×10^{-12}
06	0.0010513250	0.0010477027	1.7×10^{-10}
07	0.0011505656	0.0011459973	1.2×10^{-7}
08	0.0010497525	0.0010466045	6.8×10^{-11}
09	0.0010180373	0.0010143776	4.4×10^{-12}
obs_00	0.0013644110	0.0013595903	2.8×10^{-9}
obs_01	0.0008004016	0.0007989995	6.1×10^{-5}
obs_02	0.0017626268	0.0017539464	4.9×10^{-12}
obs_03	0.0014515488	0.0014433616	9.5×10^{-12}
obs_04	0.0015765581	0.0015672729	9.9×10^{-13}
obs_05	0.0011700848	0.0011640779	5.0×10^{-14}
obs_06	0.0010516695	0.0010485993	1.7×10^{-8}
obs_07	0.0011508833	0.0011468338	9.4×10^{-9}
obs_08	0.0010499596	0.0010463940	7.4×10^{-10}
obs_09	0.0010186799	0.0010147030	1.9×10^{-13}
all	0.001239483	0.001234030	

TABLE III: Results with a mutation rate of 15%.

Scenario	Baseline	Model-based	p value
00	0.0013684841	0.0013620713	4.3×10^{-13}
01	0.0008024467	0.0008002416	5.4×10^{-9}
02	0.0017670298	0.0017562541	2.8×10^{-14}
03	0.0014546409	0.0014450204	9.2×10^{-14}
04	0.0015801424	0.0015689822	5.8×10^{-16}
05	0.0011722525	0.0011655343	1.1×10^{-15}
06	0.0010537881	0.0010500097	2.5×10^{-10}
07	0.0011539474	0.0011497614	1.3×10^{-9}
08	0.0010518797	0.0010478246	3.4×10^{-12}
09	0.0010200588	0.0010154878	2.5×10^{-13}
obs_00	0.0013685772	0.0013636066	1.2×10^{-6}
obs_01	0.0008031774	0.0008009511	1.2×10^{-9}
obs_02	0.0017683653	0.0017560260	2.3×10^{-16}
obs_03	0.0014557098	0.0014463438	1.5×10^{-11}
obs_04	0.0015812786	0.0015700461	4.5×10^{-15}
obs_05	0.0011725839	0.0011658291	7.5×10^{-16}
obs_06	0.0010538718	0.0010503070	8.8×10^{-10}
obs_07	0.0011545737	0.0011497825	3.4×10^{-10}
obs_08	0.0010523206	0.0010480330	2.5×10^{-10}
obs_09	0.0010206659	0.0010156657	4.7×10^{-15}
all	0.001242790	0.001236389	

the mutation rate. Furthermore, the difference in performance between the two algorithms is extremely significant, as the p-values in the tables show.

To further investigate the difference in behaviour between the two algorithms, we plotted convergence curves for each mutation rate/scenario pairing. One of them, as an example, is depicted in Figure 3.[1] The figure shows number of evaluations vs. mean cost of the best solution, averaged over runs. The curves clearly show a significant performance difference between the two algorithms. Interestingly, the adaptive model-

[1]Space prevents us including all of the figures.

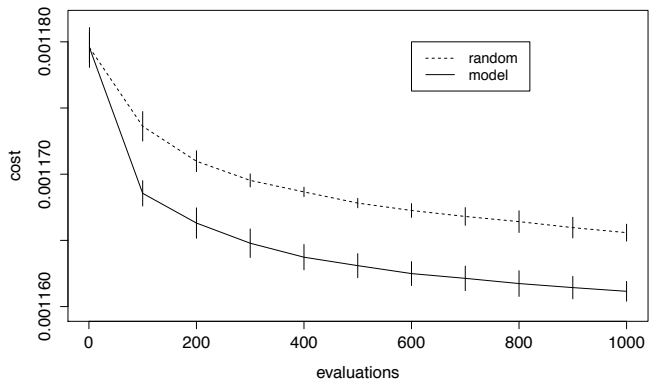


Fig. 3: Example of convergence curves for the obs_05 scenario. Both curves are averages over ten runs, and vertical bars indicate the standard deviation in cost over the runs at each point in the optimisation. The mutation rate used is 1%.

based mutation operator achieves most of its gains over the baseline algorithm early in the optimisation process; after a certain number of evaluations, the rate of cost decrease for both algorithms tends to be about the same.

We were also interested in determining if there was any clear visual difference in the ES-optimised layouts compared to the initial random layouts that the ES starts with. To this end, Figure 4 depicts layouts for Scenario obs_05 before and after application of our ES. The obstacles in this example are depicted by rectangles, and the turbine positions by points. Visual inspection shows that there is a difference between the two layouts. In particular, Figure 4(b), the optimized layout, has turbines placed closer to the edges of the layout and the border of obstacles compared to Figure 4(a), the random layout. This makes sense intuitively for two reasons. Firstly, if the layout has more turbines closer to its edge, then the interior spacing between turbines increases and this reduces the possibility of turbines within the layout interfering with each other. Similarly, placing turbines closer to the edges and obstacle borders increases the chance that a turbine's wake will spread out over an area where other turbines cannot be placed. This also reduces wake interference. Both observations may be useful for designing a more advanced turbine placement heuristic in the future.

V. Conclusion

To conclude, the novel contribution of this paper is a new mutation operator (based on predictive modelling) for the wind farm layout optimisation algorithm. This operator may be used either in conjunction with a more sophisticated search algorithm or simply with a standard ES. Evaluation shows that this novel approach is superior compared to a typical randomised mutation operator.

References

[1] H. Beyer and H. Schwefel. Evolution strategies: A comprehensive introduction. *Journal Natural Computing*, 1(1):3–52, 2002.

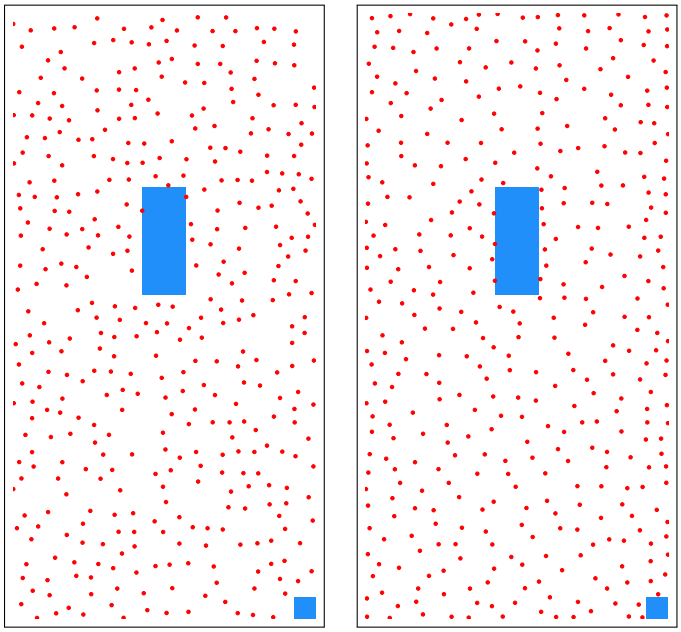


Fig. 4: Wind farm layouts examples. The examples are related to Scenario 05 and show layouts (a) before and (b) after application of the 1+1 ES with mutation rate of 1%.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
[3] A. Kusiak and Z. Song. Design of wind farm layout for maximum wind energy capture. *Renewable Energy*, 35:685–694, 2010.
[4] G. Mosetti, C. Poloni, and D. B. Optimization of wind turbine positioning in large wind farms by means of a genetic algorithm. *Journal of Wind Engineering and Industrial Aerodynamics*, 51(1):105–116, 1994.
[5] M. Samorani. The wind farm layout optimization problem. In P. Pardolas, editor, *Handbook of Wind Power Systems*. Springer-Verlag, 2013.
[6] M. Wagner, J. Day, and F. Neumann. A fast and effective local search algorithm for optimizing the placement of wind turbines. *Renewable Energy*, 51(0):64–70, 2013.
[7] D. Wilson. http://www.irit.fr/wind-competition/.
[8] D. Wilson, S. Cussat-Blanc, K. Veeramachaneni, U. O'Reilly, and H. Luga. A continuous development model for wind farm layout optimization. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 745–752, New York, NY, USA, 2014. ACM.

Acknowledgements

Thanks to D. Wilson, S. Cussat-Blanc, S. Rodrigues and K. Veeramachaneni for kindly providing the wind farm simulator and evaluation function code.