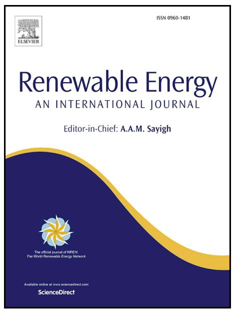

Evolutionary computation for wind farm layout optimization

Dennis Wilson, Silvio Rodrigues, Carlos Segura, Ilya Loshchilov, Frank Hutter, Guillermo López Buenfil, Ahmed Kheiri, Ed Keedwell, Mario Ocampo-Pineda, Ender Özcan, Sergio Ivvan Valdez Peña, Brian Goldman, Salvador Botello Rionda, Arturo Hernández-Aguirre, Kalyan Veeramachaneni, Sylvain Cussat-Blanc

PII: S0960-1481(18)30363-X

DOI: 10.1016/j.renene.2018.03.052

Reference: RENE 9921

To appear in: *Renewable Energy*

Received Date: 15 July 2017

Revised Date: 7 March 2018

Accepted Date: 20 March 2018

Evolutionary computation for wind farm layout optimization

Dennis Wilson[a], Silvio Rodrigues[b], Carlos Segura[d], Ilya Loshchilov[e], Frank Hutter[e], Guillermo López Buenfil[d], Ahmed Kheiri[f], Ed Keedwell[g], Mario Ocampo-Pineda[d], Ender Özcan[h], Sergio Ivvan Valdez Peña[d], Brian Goldman[i], Salvador Botello Rionda[d], Arturo Hernández-Aguirre[d], Kalyan Veeramachaneni[c], Sylvain Cussat-Blanc[a]

[a]*University of Toulosue - IRIT - CNRS UMR5505, {dennis.wilson, sylvain.cussat-blanc}@irit.fr*
[b]*Delft University of Technology, s.m.fragosorodrigues@tudelft.nl*
[c]*MIT - LIDS, kalyanv@mit.edu*
[d]*Center for Research in Mathematics, {carlos.segura, manuel.lopez, mario.ocampo, ivvan, botello, artha}@cimat.mx*
[e]*University of Freiburg, {ilya,fh}@cs.uni-freiburg.de*
[f]*Lancaster University, a.kheiri@lancaster.ac.uk*
[g]*University of Exeter, e.c.keedwell@exeter.ac.uk*
[h]*University of Nottingham, ender.ozcan@nottingham.ac.uk*
[i]*Michigan State University, brianwgoldman@acm.org*

Abstract

This paper presents the results of the second edition of the Wind Farm Layout Optimization Competition, which was held at the 22^{nd} Genetic and Evolutionary Computation COnference (GECCO) in 2015. During this competition, competitors were tasked with optimizing the layouts of five generated wind farms based on a simplified cost of energy evaluation function of the wind farm layouts. Online and offline APIs were implemented in C++, Java, Matlab and Python for this competition to offer a common framework for the competitors. The top four approaches out of eight participating teams are presented in this paper and their results are compared. All of the competitors' algorithms use evolutionary computation, the research field of the conference at which the competition was held. Competitors were able to downscale the optimization problem size (number of parameters) by casting the wind farm layout problem as a geometric optimization problem. This strongly reduces the number of evaluations (limited in the scope of this competition) with extremely promising results.

Keywords: Wind farm layout optimization, evolutionary algorithm, competition

1. Introduction

Wind farm design is a complex task and the recent trend of larger farm sizes has greatly increased demands on designers. Traditionally, a small, well-connected, land area is divided into smaller cells and turbine placement among cells is decided through a simple search algorithm with a pre-specified cost function. This function is usually limited to minimizing inter-turbine wake interferences and thus maximizing energy capture. Few approaches consider additional factors such as operation and maintenance costs, turbine costs, or cable layout.

Modern farms cover large areas and boast hundreds, and sometimes even thousands, of turbines. The layout design process is iterative, computationally expensive, burdened with global and local constraints, and ultimately controlled by subjective assessments due to the involvement of a variety of stakeholders. During each step, designers must either refine an incremental layout or propose a new layout which they have generated by incorporating new constraints. Additionally, evaluating a layout requires varied multi-disciplinary models and sub-modules that are extremely computationally expensive.

The wind farm layout optimization problem is the identification of turbine positions in a 2-D plane such that the energy capture is maximized while costs associated with a number of other factors are minimized. The energy capture for a turbine takes into account the wind scenario (wind force distribution and terrain), the turbines' power curve (power generated by the turbine in function of the wind input) and wake effects (inter-turbine interferences) [1].

When optimizing both the position and the number of turbines to build, genetic algorithms (GAs) are commonly used to optimize wind farm layout. The farm area is discretized with a grid [2, 3, 4, 5, 6, 7, 8]. The GA optimizes a binary genome in which each gene represents the presence or absence of a turbine in each cell of the grid, therefore both optimizing the number of turbines and their discrete placement. Other techniques have been evaluated using particle swarm optimization [9, 10, 11] and local modification with different optimization algorithms [12, 13]. Wilson et al. have also proposed an innovative interactive approach based on cell-based developmental model in [14]. Interested readers can find overviews of existing methods for wind farm layout optimization in Khan et al. [15] and Samorani [16].

In this paper, we report on a competition we ran at the Genetic and Evolutionary Computation Conference 2015 (GECCO 2015) during which experts from the evolutionary computation community optimized wind farm layouts. Eight teams participated and proposed innovative algorithms, all evaluated in the same context: the same wind scenarios, power curve and wake effect models. This paper presents the results of the top 4 participants and is organized as follows. Section 2 presents the rules and the framework used during the competition. Section 3 details the top 4 algorithms of the competition, which are then compared with a standard binary GA in section 4. Finally, this article discusses these algorithms and opens novel questions that could be addressed using evolutionary algorithms in this domain of research.

2. Competition rules and framework

2.1. Competition rules

Whereas the first edition of the wind farm layout optimization competition, held at the 2014 Genetic and Evolutionary Computation COnference (GECCO), consisted in only optimizing the wake free ratio (actual energy output over potential output without wake), the second edition focuses on the economical viability of the produced layouts. Layouts generated by the competitors' algorithms are evaluated in the cloud on 5 unknown wind scenarios (wind rose, layout shape and obstacles, turbine specifications, etc.) using the cost function presented below. In order to keep the computation cost acceptable, the competitors have a finite number of possible evaluations: they can only call the evaluation function 10,000 times for all 5 scenarios combined. This limited amount of evaluation credits aims to represent the CPU cost of layout evaluation and to promote efficient algorithms. This metric was preferred to CPU time because the computation was held on a shared research cluster with no exclusive access guaranteed. In order to develop their algorithms, competitors also have access to 20 known scenarios, 10 without obstacles and 10 with obstacles, all different from the ones used during the competition.

Based on the best layouts submitted, the competing algorithms are compared on each scenario to optimize the cost of energy function. The algorithms are ranked on each scenario with a point system provided in Table 1.

Position	1st	2nd	3rd	4th	5th	6th
Points	10	6	4	3	2	1

Table 1: Point system

Places below 6th receive 0 points. The winner of the competition is the competitor that has the highest number of points among the 5 scenarios.

2.2. Layout evaluation

For each layout generated, the goal of the competitors is to minimize the cost of energy f calculated as follows:

$$f = \frac{(c_t n + c_s \lfloor \frac{n}{m} \rfloor) + c_{OM} n}{(1-(1-r)^{-y})/r} * \frac{1}{8760P} + \frac{0.1}{n} \tag{1}$$

where c_t is the turbine cost, c_s is the price of a substation, m is the number of turbines per substation, r is the interest rate, y is the farm lifetime in years, c_{OM} is the operation and maintenance costs, n is the number of turbines of the layout, P is the layout's energy output. The last term of the fitness function rewards layouts with more turbines: without this term, smaller layouts would be more optimal, which would be counterproductive with regard to the optimization.

This cost function corresponds to the production price of a kilowatt: the competitors must produce the layout that minimizes the cost of energy with respect to the scenario provided. Table 2 provides the constant values used in the previous equation.

Note that a set of constraints must be fulfilled by a wind farm layout to be valid. First, all the turbines must be located inside the terrain where the wind farm will be deployed, i.e. all turbine positions must be smaller than a maximum x and y provided in each scenario. Second, the distance between turbines must be larger than a given security threshold, which is called D_{min} and in our studies was set to 8 times the turbine rotor radius.

Constant	Value
c_t	\$750,000
c_s	\$8,000,000
m	30
r	3%
y	20 years
c_{OM}	\$20,000 per year
D_{min}	308m

Table 2: Constant values used in the calculation of the cost of energy.

To evaluate the energy output of the wind farm, Kusiak's model has been used [1]. It provides the energy output P of the layout generated by the competitor optimizers.

2.3. Inter-turbine interference model

The energy capture for a turbine takes into account the following:

- *Wind Scenario*: Wind speed, v, represented as a random variable with a Weibull distribution that is a summary of wind speed at that location for a period of time. This is given by $p_v(v; c, k|\theta)$, where c and k are Weibull shape and scale parameters and θ is a wind directional bin. The wind speed distribution is different for different directional bins, θ. Additionally, wind flows from a certain direction with some probability $p(\theta)$. Together $p_v(v; c, k|\theta)$ and $p(\theta)$ are referred to as wind resource/scenario in this paper.

- *Power curve*: A function $\eta(v)$, known as a *power* curve, gives the power generated by a turbine for the given wind speed v. The power curve is dependent on the turbine make and model.

- *Wake effects*: In a particular directional bin, for a given turbine i located at x_i, y_i a number of other turbines affect the wind it experiences. This is called wake effect. Given the other turbines' locations x_j, y_j for all $j \in \{1 \dots i-1, i+1 \dots n\}$, the turbines that affect the particular turbine are determined using a *wake* model. This is documented in [1]. If a turbine located at x_i, y_i is in the wake of another turbine located at x_j, y_j in a given direction, the wind speed distribution experienced by the turbine is modified by changing the parameter c resulting in a turbine specific $c_i|\theta$. The value $c_i < c$ is reduced in proportion to the Euclidean distance between i and j.

To evaluate the energy capture, the objective function needs the *expected* value of the energy capture for a given wind resource and turbine positions. For a single turbine at position (x_i, y_i), it first determines its modified wind resource for each directional bin based on other turbine positions and then calculates its energy capture using:

$$E = \int_\theta p(\theta) \int_v p_v^\theta(v; c_i, k_i|\theta)\eta(v). \tag{2}$$

Equation 2 evaluates the overall average energy over all wind speeds for a given wind direction, and then averages this energy over all wind directions. Energy is calculated for every turbine and then summed together to give global energy capture.

2.4. Framework

In order to reduce the competitors' development efforts, we have developed an open-source API, called WindFLO, that implements the cost function (and the inter-turbine interference model) in multiple languages (C++, Java, Matlab

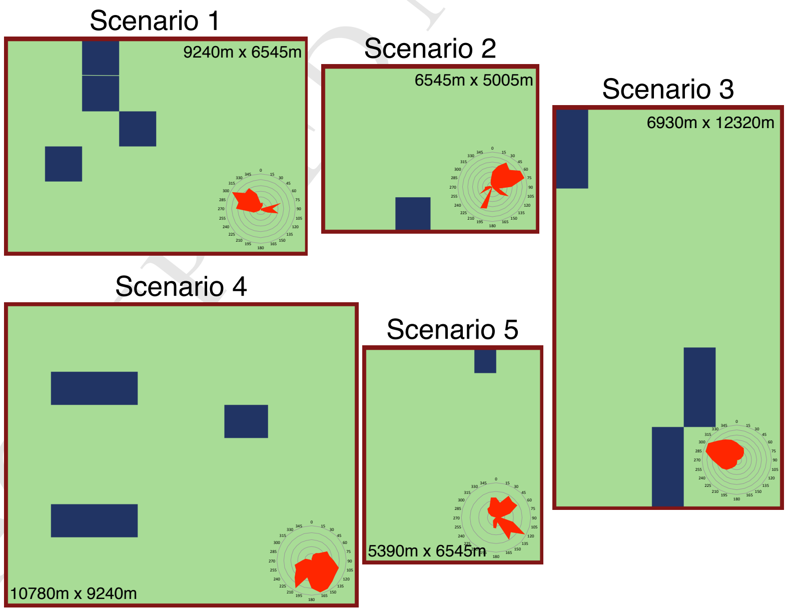

Figure 1: The 5 unknown scenarios used to evaluate the competitors' algorithms. Obstacles are represented with dark blue rectangles.

and Python)[1]. The API provides a simple GA as an example of use of the library.

2.5. Wind scenarios

Figure 1 depicts the scenarios (terrain sizes, obstacles and wind roses) used to evaluate the competitors' algorithms. Wind roses graphically represent the wind repartition in force and probability in each direction. These scenarios can be downloaded from the WindFLO repository.

[1]This API is available on github: `https://github.com/d9w/WindFLO`

3. Competitors' algorithms

The second edition of the competition received a total of 8 submissions. This section presents the top four approaches. In our opinion, they are the most relevant to the wind farm optimization community and provides the best results in term of quality of layouts obtained. These four algorithms are available in the WindFLO API presented above. These algorithms are the following:

1. *3s-MDE*: from Carlos Segura, Guillermo López Buenfil, Mario Ocampo Pineda, Sergio Ivvan Valdez Peña, Salvador Botello Rionda, and Arturo Hernández-Aguirre. The 3-Stages Memetic Differential Evolution (3s-MDE) starts by creating a surrogate model which approximates the cost function. Then, a memetic differential evolution is used to pre-optimize the model based on a geometric distortion of a layout based on rhomboids. The pre-optimized layout is then refined by locally modifying the candidate solution. The more accurate model is used only to evaluate solutions with a promising behaviour in the surrogate model.
2. *CMA-ES*: from Ilya Loshchilov and Frank Hutter, this second approach uses the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to optimize a layout described by 5 variables: scale (horizontal and vertical), shift from the origin, rotation, and shift from a given location.
3. *SSHH*: from Ahmed Kheiri and Ed Keedwell, the Sequence-based Selection Hyper-Heuristic (SSHH) approach discretized the layout and then a solution is represented by three integer variables that corresponds to the distance between neighbouring turbines and a shift factor. A hidden Markov model produces a sequence of low level heuristics which create the final layout.
4. *GM*: from Brian Goldman, the Goldman Method (GM) presented in this paper uses a pair of lattice vectors to calculate turbine locations. It also uses the cost of substation, which is larger than the turbine cost itself, to leverage the size of the evaluated layouts. A deterministic best-improvement local search method is then used to optimize the lattice vectors.

The source codes of these four algorithms are available on the competition github: `https://github.com/d9w/WindFLO`. We now discuss these approaches in turn.

3.1. Memetic Differential Evolution with Surrogate Model and Ad-hoc Improvements (3s-MDE)

In this section we present a three-stage optimizer — see Algorithm 1 — that combines a memetic differential evolution (MDE) with a novel representation of candidate solutions, a surrogate model and ad-hoc improvements. This method is referred to as 3S-MDE in the rest of the paper. A detailed flowchart of the scheme describing the three stages is given in Fig. 2. In this flowchart, the boxes that perform evaluations in the real simulator — in contrast to those that use the surrogate model — are shown in gray. The following sections describe each stage of this optimizer.

Algorithm 1 Memetic DE with Surrogate Model and Ad-hoc Improvements

```
1: Stage 1: Create a Surrogate Model $SM$.
2: Stage 2: Apply a memetic DE/rand/1/bin using $SM$ with a representation based
   on rhomboids.
3: Stage 3: Apply ad-hoc improvements to the best solution obtained in Stage 2.
```

3.1.1. First Stage: Construction of a Surrogate Model

One of the main difficulties when dealing with complex engineering problems is that evaluating a solution is usually quite an expensive process. Surrogate models can be used to alleviate this difficulty [17]. Different ways of building surrogate models have been proposed. Problem-dependent models rely on ad-hoc knowledge of the problem, whereas with problem-independent models, machine learning methods are usually employed [18].

For this optimizer, a problem-dependent surrogate model is built to approximate the energy generated by a given set of turbines. In order to construct the surrogate model, a set of simple candidate solutions where only two turbines are placed in the wind farm are evaluated using the original evaluator. Additionally, one evaluation with a single turbine is carried out to estimate the maximum energy that it can generate. For each evaluation with two turbines, the penalty on the energy produced in both turbines with respect to the case where only one turbine is placed in the wind farm is calculated. Specifically, the following set of distances are checked: D_{min}, $1.5D_{min}$, $2D_{min}$, $2.5D_{min}$, $3D_{min}$ and $3.5D_{min}$, where D_{min} represents the minimum admissible distance between turbines. For the case in which the distance between turbines is set to D_{min}, 720 equidistributed different angles are checked, whereas in the remaining cases, 360 different angles are used.

The surrogate model uses the saved data to approximate the energy generated by layouts that use any number of turbines. The model assumes that the energy generated by one turbine is not influenced by any other turbines placed further away than $3.5D_{min}$. Thus, for each turbine, the set of conflicting turbines — those at a distance not greater than $3.5D_{min}$ — is detected and the negative influence is estimated based on the saved data. Specifically, if for a given turbine there exists another turbine at a distance D and angle γ, then the four surrounding cases — combining the nearest distances and angles — are identified and linear interpolations are used to estimate the negative influence. Note that the computational complexity of identifying the surrounding cases with values higher and lower than D and γ is constant, so this process is quite fast. The total energy generated by a given turbine is the estimate of the maximum attainable energy minus the sum of the estimated negative influences. The total energy estimated for a given layout is the sum of the energies estimated for each turbine.

3.1.2. Second Stage: Memetic Differential Evolution

In the second stage, one of the most well-known variants of DE is applied. Specifically, the DE/rand/1/bin is used [19]. This variant has the property of

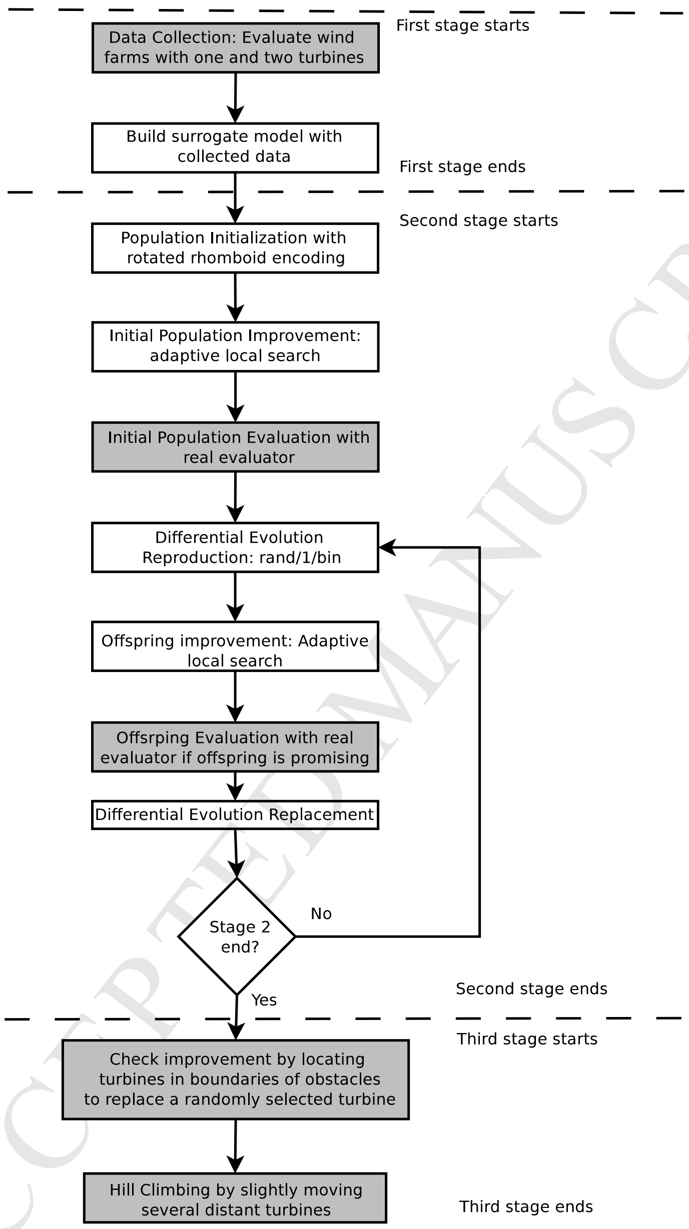
First stage starts
Data Collection: Evaluate wind farms with one and two turbines
Build surrogate model with collected data
First stage ends
Second stage starts
Population Initialization with rotated rhomboid encoding
Initial Population Improvement: adaptive local search
Initial Population Evaluation with real evaluator
Differential Evolution Reproduction: rand/1/bin
Offspring improvement: Adaptive local search
Offsrping Evaluation with real evaluator if offspring is promising
Differential Evolution Replacement
Stage 2 end?
No
Yes
Second stage ends
Third stage starts
Check improvement by locating turbines in boundaries of obstacles to replace a randomly selected turbine
Hill Climbing by slightly moving several distant turbines
Third stage ends

Figure 2: 3s-MDE Flowchart

being more explorative than schemes that apply the "best" strategy. This feature was vitally important due to the small population size (N) used in the runs. In the current MDE implementation, a special initialization of the population is used. First, 200 individuals are created; then, the best N individuals

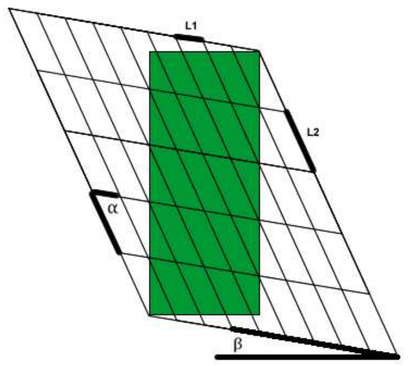

Figure 3: Encoding of Individuals in MDE

are selected to form the initial population.

One of the keys to designing proper Evolutionary Algorithms (EAs) is the selection of a suitable encoding of the individuals [20]. Directly encoding the coordinates of the turbines yields too large a search space, thus an alternative encoding was adopted. Specifically, a solution is encoded with four real-valued parameters, which establish the geometry of a possibly rotated rhomboid (see Figure 3): $L1$ and $L2$ are the lengths of the sides of the rhomboid; α is the angle formed between two sides of the rhomboid; finally, β rotates the whole rhomboid. This rhomboid is used to establish the positions where the turbines are deployed, which is illustrated in Figure 3. Specifically, the given scenario — green zone — is covered with a set of rhomboids with the specified geometry and the turbines are positioned at the vertexes of these rhomboids. One of the vertexes of the rhomboids is placed in the bottom-left corner of the wind farm. Since some turbines might be too close to one another, a repairing method is used to remove conflicting turbines. First, the turbines are randomly shuffled. Then, each turbine is sequentially examined and is accepted only if it does not conflict with any previously accepted turbine, i.e. it is accepted if the distance to any already selected turbine is not smaller than D_{min}. Additionally, when a vertex is inside an obstacle, no turbine is located in such a vertex.

In order to improve upon the intensification capabilities, DE was integrated with an adaptive local search, which is applied both in the creation of the initial population and after the creation of the offspring. The number of evaluations used in each application of the local search is set with the local search budget (LSB) parameter. The adaptive search starts with a given step size (SS) for each parameter which is progressively decreased linearly with the aim of increasing the intensification capabilities. The reduction is calculated in such a way that by the end of the local search the resulting value is 0. When each neighbor is created, any of the four parameters is changed with a probability equal to 0.25. Specifically, they are modified by summing or subtracting the step size corresponding to the iteration. Note that applying the real evaluator in the local search would be very expensive. As a result, the adaptive local search uses the surrogate model and only the best solution obtained after the application of each local search might be evaluated with the original evaluator. Specifically, the obtained solution is evaluated with the original evaluator only if it is not worse than 1.20 multiplied by the best so far obtained fitness. Thus, the surrogate model is also used to filter out poor solutions, which is a typical use of surrogates.

3.1.3. Third Stage: Ad-hoc Improvement

The last stage starts from the best solution identified by the memetic DE and applies two methods to further improve it. Note that in this last phase, the surrogate model is not used because some minor modifications are made to the candidate solutions, thus making it impossible to properly measure the promising behavior of these changes with the surrogate model.

The first method is applied only in the scenarios where obstacles are included. First, given a budget B for evaluations, $\frac{B}{2}$ potential locations are identified. Specifically, at each edge of the obstacles, $\frac{B}{N_{OBS}*4*2}$ locations are equidistributed, where N_{OBS} is the number of obstacles. Then, for each selected location, a new solution is evaluated that considers the establishment of a turbine in that position and the removal of a randomly selected turbine. The modification is maintained only if the fitness is improved.

Finally, the remaining evaluations are used to apply a local search. In the initial version of the local search, neighbors were generated by slightly moving a single turbine. Specifically, a maximum step (MS) was established — here set to 6.25 units — and in order to generate a neighbor, a turbine was moved S units, where S was a randomly generated number between 0 and MS. The direction of movement was also randomly generated. Given that the number of evaluations is very restricted, instead of moving a single turbine, the local search moves several turbines simultaneously, following the same method explained above. Distant turbines are selected using the process below. First, a safety distance (SD) is established to $6 \times D_{min}$. Then, a turbine present in the current solution is randomly selected and a grid of square cells with a side length equal to SD is set up in the wind farm in such a way that the center of a cell is in the position of the selected turbine. Then, for each cell, the turbine that is closest to the center is selected. In this way, a set of distant turbines is generated. The new candidate solution is created by moving each selected turbine. Then, the energy generated in each cell is calculated and the cells where the amount of energy generated is increased are considered to be promising moves. Finally, the movements involving unsuccessful cells are undone and the new candidate solution is reevaluated. If the fitness of the new solution improves the original solution, it is accepted. This process is repeated until the budget for the number of evaluations is exhausted.

3.1.4. Parameterization

3S-MDE needs a set of parameters, which were tuned with the set of instances given in the first part of the contest. Table 3 provides the parameters used in this competition.

Parameter name	Notation	value
Population size	N	10
Crossover rate	CR	0.9
Mutation scale factor	F	Gaussian dist. $\mathcal{N}(0.5, 0.3)$
L1 Step size	$L1$-SS	125
L2 Step size	$L2$-SS	125
α Step Size	α-SS	22.5 degrees
β Step Size	β-SS	22.5 degrees
Local search budget (Initialization)	I-LSB	15000
Local Search budget (Offspring)	O-LSB	5000
Criterion to pass to third stage	N/A	90% of Evaluations

Table 3: Parameters of the 3s-mde method.

3.2. CMA-ES for layout optimization (CMA-ES)

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [21] is the method of choice when dealing with hard black-box optimization problems which are nonlinear, multimodal and/or noisy. The algorithm is invariant to rank-preserving transformations of the objective function and affine transformations of the search space which makes it parameter-less in practice. CMA-ES has repeatedly demonstrated good performance at various platforms for comparing continuous optimizers such as the Black-Box Optimization Benchmarking (BBOB) workshop [22, 23, 24] and the Special Session at Congress on Evolutionary Computation [25, 26]. Thus, CMA-ES seems naturally suitable for the wind farm layout optimization problem especially when a low-dimensional parameterization of the design problem is considered.

In this section, we discuss the application of CMA-ES to the wind farm layout optimization problem. In a nutshell, CMA-ES works as follows. In its iteration t, a mean m^t of the mutation distribution (which can be interpreted as an estimation of the optimum) is used to generate λ candidate solutions $x_k \in \mathbb{R}^n$ by adding a random Gaussian mutation defined by a (positive definite) covariance matrix $C^t \in \mathbb{R}^{n \times n}$. The k-th sampled candidate solution x_k^t is then defined as:

$$x_k^t \sim \mathcal{N}\left(m^t, {\sigma^t}^2 C^t\right) = m^t + \sigma^t \mathcal{N}\big(0, C^t\big)\,, \tag{3}$$

where σ^t is a mutation step-size. These λ solutions then should be evaluated on an objective function f. The old mean of the mutation distribution is stored in m^t and a new mean m^{t+1} is computed as a weighted sum of the best μ parent individuals selected among the λ generated offspring individuals. The algorithm adapts m^t and C^t in order to increase the likelihood of the successful sampling steps to appear again and to proceed towards the optimum.

Since we optimize both the number of turbines and their positions, the direct representation of turbine positions as variables would lead to a large-scale

Parameter	Notation	Optimized
Layout height	H	No (input)
Layout width	W	No (input)
Turbine radius	R	No (input)
Layout scale factor	h_1	No, fixed to 4
Interturbine scale on x-axis	h_2	No, fixed to 4
Interturbine scale on y-axis	h_3	No, fixed to 4
Interturbine distance on x-axis	x_1	Yes
Interturbine distance on y-axis	x_2	Yes
Rotation angle	x_3	Yes
Shift on x-axis	x_4	Yes
Shift on y-axis	x_5	Yes

Table 4: Parameters optimized with CMA-ES and constants used in this approach.

optimization problem which is often intractable [13]. Similarly to other algorithms presented in this paper, we therefore parameterize the search space of wind-farm layouts with only a few variables. We use 5 continuous variables to i) fill a rectangular grid of turbines, ii) shift it to the origin, iii) rotate it, and then iv) shift it to some location. Finally, all turbines which violate the constraints (e.g., obstacles, target size of the layout) are removed from the grid.

We parameterize the original optimization problem as follows:

STEP 1 The initial rectangular grid is selected to be $h_1 = 4$ times larger than the maximum size $(W; H)$ of the target layout to take into account its rotation in STEP 2. The minimum distance between turbines on the x-axis is set by $D_x = D_{min} + (0.2x_1)^{h_2} \times (W - D_{min})$, where $D_{min} = 8R$ and h_2 is a hyperparameter set to 4. Similarly, the distance on y-axis is set by $D_y = D_{min} + (0.2x_2)^{h_3} \times (H - D_{min})$. Thus, the total number of turbines is $(\lfloor h_1 \times W/D_x \rfloor + 1)(\lfloor h_1 \times H/D_y \rfloor + 1)$.

STEP 2 Shift the grid to the origin by substracting $(h_1 \times W/2; h_1 \times H/2)$ and rotate turbine coordinates by an angle $\theta = -\pi + 2\pi x_3$.

STEP 3 Shift the rotated grid by $((0.5 + 0.2x_4) \times W; (0.5 + 0.2x_5) \times H)$.

STEP 4 Remove all infeasible turbines.

Table 4 summarizes the various constants used as well as the five variables x_1 to x_5 optimized by CMA-ES. All these 5 variables are constrained to be in $[0, 1]$ and are optimized with the active CMA-ES [27, 28] in its default settings and with $m^0 = [0.5]^5$ and $\sigma^0 = 0.3$.

3.3. A Sequence-based Selection Hyper-heuristic (SSHH)

In contrast to the other techniques, hyper-heuristics operate at the level above (meta-)heuristics such as EAs. Selection hyper-heuristics are high level control methods that mix and manage a predefined set of low level heuristics for solving hard computational problems. Hence, online learning is a crucial component of such methods which are capable of discovering the appropriate combinations of low level heuristics yielding an improved overall performance. More on different types of hyper-heuristics, including an overview of different algorithmic components and designs, and their applications can be found in [29].

Traditionally, a selection hyper-heuristic employs two methods, consecutively: a *heuristic selection* method to choose a suitable low level heuristic (move operator) which is applied to a candidate solution, and then a *move acceptance* method to decide whether to accept or reject the newly generated solution. This study presents a sequence-based selection hyper-heuristic (SSHH) consisting of a learning sequence of heuristics selection method and an adaptive threshold move acceptance method to solve the wind farm layout optimization problem.

3.3.1. Representation and overall search

In order to pick the best locations for a set of turbines, a given region is split into a grid consisting of a certain number of rows and columns with neighbouring cells having an interval of size (0.1×TurbineRadius) in between them. A given solution is represented using three integer variables, X, Y and S, where $M \leq X < maxCols$, $M \leq Y < maxRows$, and $0 \leq S$; M is the minimum allowed distance between neighbouring turbines, $maxRows$ and $maxCols$ are the maximum number of rows and columns, respectively (see Figure 4). Based on the values of those variables, a binary matrix is formed by spreading neighbouring turbines with a separating distance of X cells at each row and a separating distance of Y cells at each column. A cyclic shift is performed at each row starting from the second row. The S parameter is used to reduce the wake effect from a neighbouring turbine and deals with differing wind directions by shifting wind turbines. Algorithm 1 provides the pseudocode of how the solution is constructed.

Algorithm 1: The pseudocode of how a solution is constructed

```
input  : scenario, $X$, $Y$ and $S$
output: grid
1 for $i \leftarrow 0; i < maxCols; i \mathrel{+}= X$ do
2     for $j \leftarrow 0; j < maxRows; j \mathrel{+}= Y$ do
3         $l = (i + (j/Y) * S) \% maxCols$;
4         if $scenario$.isValidLocation($l, j$) then
5             grid.insertTurbineAtLocation($l, j$);
6         end
7     end
8 end
```

SSHH performs the search under an iterative framework maintaining the best solution detected so far. Solutions are always made feasible by removing the invalid turbines which are placed on obstacles, before evaluation.

3.3.2. Low level heuristics

The low level heuristics (LLHs) used in this work are parameterized low level heuristics. Assume that $H(V_1, p, V_2)$ is a heuristic, where p has one of

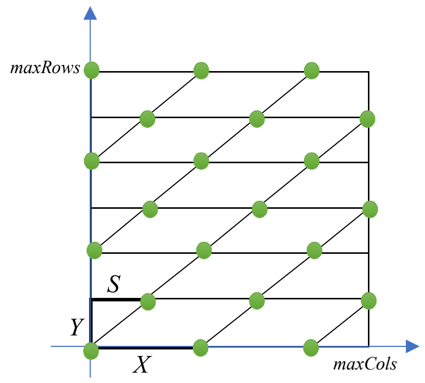

Figure 4: Encoding of Individuals in SSHH

three values: 1, rand[1,10] or rand[10,79]; and rand[L, U] is a random integer in [L, U] and this low level heuristic modifies (increases or decreases) the value of the variable V_1 by p, and then with a probability of 0.30 resets the value V_2. SSHH controls the following low level heuristics during the search process:

- **LLH0:** Apply $H(X, p, S)$
- **LLH1:** Apply $H(Y, p, S)$
- **LLH2:** Apply $H(S, p, X)$, then with a probability of 0.30 reset the value of Y.
- **LLH3:** increase or decrease the value of X by p; then update p and apply $H(Y, p, S)$.
- **LLH4:** increase or decrease the value of X by p; then update p and increase or decrease the value of Y by p; and finally update p and increase or decrease the value of S by p.

3.3.3. Heuristic sequence selection and move acceptance

The selection component forms and applies a sequence of low level heuristics to a candidate solution as a single operation, thereby generating super-heuristics through the combination of more than one low level heuristic. This online learning method analyses and produces sequences of heuristics during the search process using a hidden Markov model, in which the hidden states are low level heuristics (LLHs) and observations are sequence-based acceptance strategies (AS). We make use of two matrices: a transition probability matrix to determine the movement between states and an emission probability matrix to determine whether the sequence of heuristics that has been constructed will be applied to a candidate solution or that sequence will be extended by including another LLH. As explained earlier, each heuristic is associated with a parameter, p influencing its behaviour. Therefore, an additional emission probability matrix is implemented to set the value of p. A detailed description of this algorithm can be found in [30, 31, 32]. The threshold move acceptance method accepts all improving moves by default. However, the non-improving moves are accepted only if the cost of the new solution is less than or equal to a threshold which changes with respect to the best solution during the search process. The threshold is always set to the (1.01) of the cost of the best solution found so far, whenever a non-improving move occurs.

3.4. The Goldman Method (GM)

3.4.1. Representation

There are two straightforward approaches to representing turbine locations in the field, both of which have significant limitations.

The first method would be to enforce a maximally compact two-dimensional mesh, with each mesh point representing a possible turbine. This method handles obstacles gracefully as invalid mesh points can be removed before beginning search. As search focuses on which mesh points to include as turbines, the total number of turbines in the field is easy to manipulate. However, the mesh structure does not allow turbines to orient themselves with the wind. For example, it is possible that the wind is blowing parallel to one of mesh axes, resulting in all turbines in a row (or column) interfering with each other.

The second method would be to specify some number of turbines and then search their possible locations in the field. While this can overcome the problem of interference, it massively increases the search space. Furthermore, it now becomes challenging to determine the correct number of turbines to use and to avoid obstacles.

In an attempt to gain the best of both techniques, while simultaneously reducing the search space size, the Goldman method represents a layout as two vector lattice. In this form, search is performed over the space of a pair of two-dimensional vectors. These vectors are converted to a layout by placing a turbine at all integer linear combinations of the two lattice vectors. As with the mesh representation any invalid turbine is removed, resulting in simple obstacle handling. Yet unlike that method turbines can orient at any angle to each other, with a flexible amount of space between each, to avoid interference. To aid search, the Goldman method encodes the two lattice vectors as (angle, magnitude) pairs.

3.4.2. Number of Turbines

A significant cost of any layout is how many substations must be purchased. The cost of energy function includes the term $c_s * \lfloor \frac{N}{M} \rfloor$. The cost of a substation, c_s, is an order of magnitude larger than the cost of a turbine. As a result the cost of energy is often minimized when $N \bmod M = M - 1$ as this uses the maximum number of turbines before a new substation must be purchased.

The Goldman method leveraged this knowledge when evaluating layouts. Along with calculating the global cost of energy, each evaluation provides the user with information about the “fitness” of individual turbines in the layout.

The Goldman method used this information to reduce each tested layout to the nearest complete substation by naïvely removing the least fit turbines. The reduced layout is then evaluated, meaning that every lattice requires at most two evaluations.[2] The lattice's cost of energy is then considered to be whichever version was cheaper.

3.4.3. Search Method

Given the limited evaluation budget, the Goldman method utilizes a deterministic best-improvement local search method to explore the space of lattice vectors. To simplify this process and further reduce the search space, the four variables (two angles and two magnitudes) which encode the lattice were discretized. Angles were restricted to 36 intervals spaced $\pi/18$ apart. Magnitudes ranged from the minimum allowable interval between turbines up to 5 times that size in 64 evenly spaced steps.

To perform local search, the algorithm began from a lattice with the vectors perpendicular to each other, one using the minimum magnitude and the other using the middle of the magnitude range. All alternatives to each variable were then tested sequentially, such that only the best changing improvement to a variable is made before continuing to the next variable. This process continues until no single variable can be changed to make the lattice better. To improve rotational symmetry, this process is run starting with the long vector parallel to the x-axis, and then again parallel to the y-axis. Duplicate work is prevented by caching the quality of each discrete lattice.

[2]Some lattices create layouts where $N \bmod M = M - 1$

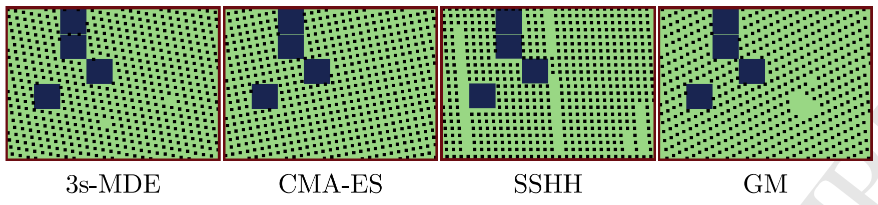

Figure 5: Best layouts obtained by the 4 competitors on scenario 1.

4. Competition results

4.1. Competition results

The algorithms presented in the previous section, in addition to four others not described in this paper, were run on the 5 scenarios presented in the competition rules section. As mentioned above, the competitors were given a budget of 10,000 evaluations to split between the 5 scenarios for computational cost reasons. Table 5 provides the costs of energy fitness of each algorithms. As a basis of comparison, we have compared the results to a genetic algorithm. GAs have been used many times in this domain and offer a familiar and standard benchmark against other algorithms from the field of evolutionary computation. For this problem, a GA optimizes a binary genome that decides whether or not a turbine is located in each grid of a discretized layout. The fitness function used to evaluate each layout is the one presented in equation 1. The GA is set up with a population size of 20 individuals, a 4-player tournament selection with elitism, 5% mutation and 40% crossover. Mutating a layout consists in switch a Boolean of the grid and crossing is a standard one-point crossover. The GA is run for 50 generations, which represents 2000 evaluation per scenario.

Table 7 provides both the number of turbines and of substations of the best layouts obtained by the different competitors on the different scenarios. It is worth noting that obtained layouts are of comparable size in terms of number of turbines for each scenario. Also, the algorithms, either naturally or due to specific strategies, limit the number of turbines to a value very close to the constraints of the substations. When evaluating the cost of energy function, the cost of building a substation is subsequently greater than the total price of the turbines. Therefore, the number of substations, even though not directly highlighted in the problem description, is of primary importance. Figure 5 depicts

Scenario	3s-MDE	CMA-ES	SSHH	GM	GA
1	$\mathbf{1.16_{4422E^{-3}}}$	$1.17_{2731E^{-3}}$	$1.18_{1129E^{-3}}$	$1.18_{5466E^{-3}}$	$1.26_{9266E^{-3}}$
2	$\mathbf{1.00_{929E^{-3}}}$	$1.02_{9998E^{-3}}$	$1.03_{9825E^{-3}}$	$1.04_{4906E^{-3}}$	$1.15_{8464E^{-3}}$
3	$\mathbf{6.26_{867E^{-4}}}$	$6.30_{916E^{-4}}$	$6.40_{241E^{-4}}$	$6.49_{096E^{-4}}$	$6.91_{265E^{-4}}$
4	$6.53_{861E^{-4}}$	$\mathbf{6.53_{56E^{-4}}}$	$6.66_{205E^{-4}}$	$6.64_{341E^{-4}}$	$7.18_{626E^{-4}}$
5	$\mathbf{1.14_{2309E^{-3}}}$	$1.15_{2661E^{-3}}$	$1.16_{7168E^{-3}}$	$1.16_{033E^{-3}}$	$1.26_{9238E^{-3}}$

Table 5: Cost of energy, compared to the state-of-the-art layout optimization (binary GA).

Scenario	3s-MDE	CMA-ES	SSHH	GM	GA
1	10	6	4	3	-
2	10	6	4	3	-
3	10	6	4	3	-
4	6	10	3	4	-
5	10	6	3	4	-
Total	46	34	18	17	-
Rank	1st	2nd	3rd	4th	-

Table 6: Competition results in points and ranking. Note that GA was not part of the competition and is therefore not ranked. The results of other competitors are not presented in this table.

Scenario	#Turbines				#Substations			
	3s-MDE	CMA-ES	SSHH	GM	3s-MDE	CMA-ES	SSHH	GM
1	539	539	561	479	18 (0)	18 (0)	19 (-8)	16 (0)
2	388	239	329	324	13 (-1)	8 (0)	11 (0)	11 (-5)
3	899	804	809	680	30 (0)	27 (-5)	27 (0)	23 (-9)
4	929	929	958	898	31 (0)	31 (0)	32 (-1)	30 (-1)
5	359	358	348	356	12 (0)	12 (-1)	12 (-1)	12 (-3)

Table 7: Number of turbines and of substations of the best layouts obtained by the competitors on each scenarios. In the substation part of the table, the value parenthesis represents the number of missing turbines in order to have all substations fully used (30 turbines per substation).

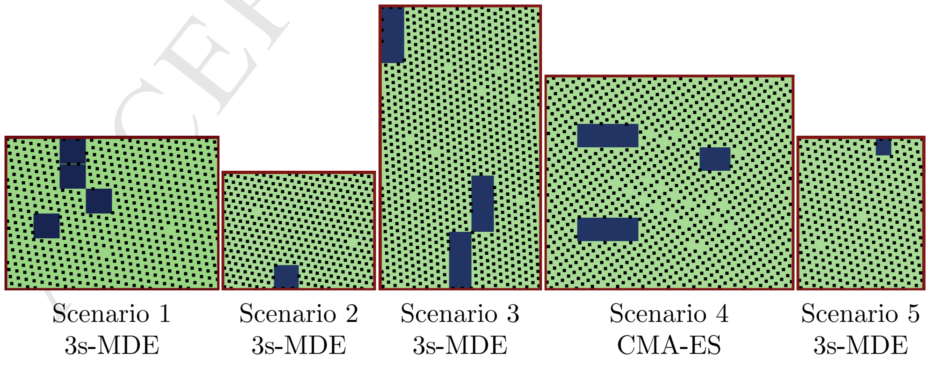

Figure 6: Best layouts obtained on the 5 scenarios.

the four layouts obtained by the four competitors for scenario 1. We can note similarities of the geometric arrangement strategies used by the competitors, albeit with different approaches, to compress the number of variables induced by the problem to a lower amount. They mostly operate on a transformation (shift, scale and rotation) of a grid layout prior to a removal of turbines that violate constraints (obstacles) and local deletion for layout refinement. Figure 6 presents the best layouts across all competitors obtained for each scenario. Once again this geometric arrangement strategy appears on the obtained layout with various rotations and shifts depending on the terrain characteristics.

4.2. Convergence comparison

Figure 7 shows the convergence curves of the different algorithms. They represent the competitor algorithms' best evaluations. Each algorithm curve stops at the final evaluation. As an example, the GA is always used with a fixed 2000 evaluation steps for each scenario.

First, the 3s-MDE optimization strategy is noticeable on the plots: whereas good fitness values appear as soon as the first evaluation in other approaches, 3s-MDE only has competitive fitness values after 1260 evaluations. This corresponds to the duration needed by the algorithm to build the surrogate model. During this initial phase, the algorithm evaluates layouts with only two turbines, which generate very poor layouts (not represented in the plot for visualization purposes).

Secondly, it can be noticed that all other algorithms converge in few iterations (approximately 300 evaluations) to a very good solutions before starting local optimization. Even though 3s-MDE needs more evaluation steps to produce a good layout, the initial 1260 evaluations are only made with 2 turbines which is costless in comparison to layouts with hundreds to thousands of turbines. Therefore, we can argue that the layouts can be presented to the farm designer very early in the optimization loop across all presented methods.

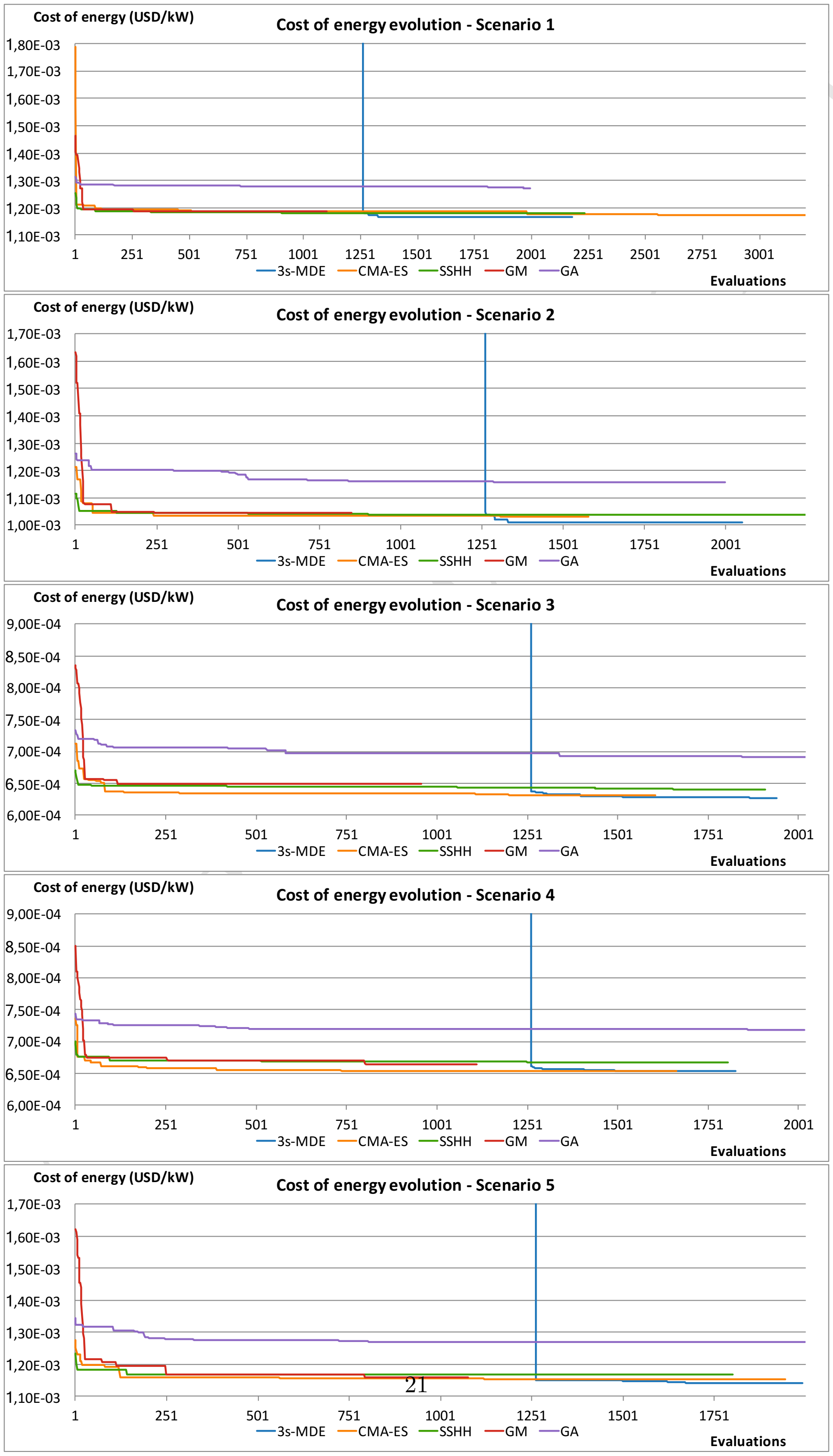

Figure 7: Comparison of the convergence profile of the different algorithms on the 5 scenarios.

5. Conclusion

This paper presented the results of the 2015 competition on wind farm layout optimization. With this event, we were able to propose innovative algorithms to optimize large wind farms with a strong computational constraint. Thanks to this competition, we were also able to compare these approaches with state-of-the-art algorithms and observe the potential improvement of the optimization algorithms used to generate the wind farm layouts.

This competition also provides a framework to compare future algorithms with existing ones on a comparative basis. The competition framework is freely available in multiple programming languages (C++, Java, Matlab and Python) with a set of randomly generated wind scenarios.

Because solutions were obtained with acceptable computational costs in this competition, we can now imagine targeting new optimization objectives. The 3D structure of the terrain, and/or heterogeneous wind distribution within the terrain, heterogeneous wind turbines with different height, width and power curves could be considered. In this competition, cable and road networks were not taken into account, but these are of great importance for the initial investment to build the wind farm. They could be added to the framework and the cost of energy function in order to be addressed by the optimization algorithms.

One of the main points we learned from this competition is that the algorithms proposed by the competitors mainly work on optimizing very few parameters in comparison to state-of-the-art algorithms; instead of optimizing the Cartesian coordinates of individual turbines, it seams preferable to optimize geometric parameters. By doing so, the complexity of the search space is drastically reduced, leading to very acceptable optimization time, even for very large wind farms. These geometric parameters allow for continuous turbine placement over the entire grid, which is clearly advantageous over the discrete grid used by the GA and previously in the literature [15]. Furthermore, these parameters could be exposed to human wind farm designers to allow them to understand the optimal placement of turbines based on the wind scenario and the constraints given to the optimization. Understanding optimized grids such as Figures 3 and 4 would allow human wind farm designers the flexibility of choosing turbine placement while still benefiting from an optimized energy cost.

Beyond demonstrating the utility of reducing this problem to an optimization of geometric parameters, the successful use of a surrogate model in this problem is novel and significant. As a surrogate could be built using gathered wind data a single time before layout design starts, the computational load of optimization during design can be greatly reduced. Surrogates could be used while considering new constraints to allow for more rapid design, and then finally rebuilt once constraints are more firmly understood. While this was not the focus of this competition, it was encouraging to see a surrogate model achieve such promising results.

This competition also encouraged the use of intelligent stop criteria by allowing competitors to develop a strategy to best allocate their evaluation over the five scenarios. The stop criterion of the optimization is still an open question for evolutionary algorithms and is a difficult one to address: even if evolutionary algorithms have been proven to converge to the optimal solution [33], it is impossible to determine when and even if the current best solution is a local optimum or the global one. This is due to the size of search space explored in this kind of problem. However, transformations on the search space can reduce its complexity, and the convergence of most algorithms here suggests that simpler search space created by optimizing geometric parameters allows for a more natural stop condition. Furthermore, we imagine the integration of this type of optimization into wind layout software will be part of an iterative design process, as numerous factors, including human design, come into play during wind farm design. The optimization process could be run for a desired number of steps or amount of computational time before being further reviewed or modified, matching the needs of human designers and mitigating the issue of a stop criterion.

Acknowledgments

D. Wilson is supported by ANR-11-LABX-0040-CIMI, within programme ANR-11-IDEX-0002-02. C. Segura acknowledges the financial support from CONACyT through the “Cienca Básica” project no. 285599. The work of A. Kheiri and E. Keedwell was funded by EPSRC grant no. EP/K000519/1.

References

[1] Andrew Kusiak and Zhe Song. Design of wind farm layout for maximum wind energy capture. *Renewable Energy*, 35(3):685–694, 2010.

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

[3] SA Grady, MY Hussaini, and Makola M Abdullah. Placement of wind turbines using genetic algorithms. *Renewable Energy*, 30(2):259–270, 2005.

[4] Hou-Sheng Huang. Distributed genetic algorithm for optimization of wind farm annual profits. In *Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on*, pages 1–6. IEEE, 2007.

[5] Chunqiu Wan, Jun Wang, Geng Yang, Xiaolan Li, and Xing Zhang. Optimal micro-siting of wind turbines by genetic algorithms based on improved wind and turbine models. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 5092–5096. IEEE, 2009.

[6] Alireza Emami and Pirooz Noghreh. New approach on optimization in placement of wind turbines within wind farm by genetic algorithms. *Renewable Energy*, 35(7):1559–1564, 2010.

[7] Sedat Şişbot, Özgü Turgut, Murat Tunç, and Ünal Çamdalı. Optimal positioning of wind turbines on gökçeada using multi-objective genetic algorithm. *Wind Energy*, 13(4):297–306, 2010.

[8] Chang Xu, Yan Yan, De You Liu, Yuan Zheng, and Chen Qi Li. Optimization of wind farm micro sitting based on genetic algorithm. *Advanced Materials Research*, 347:3545–3550, 2012.

[9] Chunqiu Wan, Jun Wang, Geng Yang, and Xing Zhang. Optimal micro-siting of wind farms by particle swarm optimization. In *Advances in swarm intelligence*, pages 198–205. Springer, 2010.

[10] Souma Chowdhury, Jie Zhang, Achille Messac, and Luciano Castillo. Unrestricted wind farm layout optimization (uwflo): Investigating key factors influencing the maximum power generation. *Renewable Energy*, 38(1):16–30, 2012.

[11] Kalyan Veeramachaneni, Markus Wagner, U-M O'Reilly, and Frank Neumann. Optimizing energy output and layout costs for large wind farms using particle swarm optimization. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–7. IEEE, 2012.

[12] Markus Wagner, Jareth Day, and Frank Neumann. A fast and effective local search algorithm for optimizing the placement of wind turbines. *Renewable Energy*, 51(0):64 – 70, 2013.

[13] Markus Wagner, Kalyan Veeramachaneni, Frank Neumann, and Una-May O'Reilly. Optimizing the layout of 1000 wind turbines. *European Wind Energy Association Annual Event*, pages 205–209, 2011.

[14] Dennis Wilson, Sylvain Cussat-Blanc, Kalyan Veeramachaneni, Una-May O'Reilly, and Hervé Luga. A continuous developmental model for wind farm layout optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 745–752. ACM, 2014.

[15] Salman A Khan and Shafiqur Rehman. Iterative non-deterministic algorithms in on-shore wind farm design: A brief survey. *Renewable and Sustainable Energy Reviews*, 19:370–384, 2013.

[16] Michele Samorani. The wind farm layout optimization problem. In *Handbook of wind power systems*, pages 21–38. Springer, 2013.

[17] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61 – 70, 2011.

[18] Alexander E.I. Brownlee, John R. Woodward, and Jerry Swan. Metaheuristic design pattern: Surrogate fitness functions. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1261–1264, New York, NY, USA, 2015. ACM.

[19] K. Price, R.M. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization.* Natural Computing Series. U.S. Government Printing Office, 2005.

[20] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing.* Natural Computing Series. Springer, 2003.

[21] N. Hansen, S.D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[22] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report 2009/21, Research Center PPE, 2010.

[23] A. Auger, S. Finck, N. Hansen, and R. Ros. BBOB 2010: Comparison Tables of All Algorithms on All Noiseless Functions. Technical Report RR-7215, INRIA, 2010.

[24] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Bi-population CMA-ES agorithms with surrogate models and line searches. In *Genetic and Evolutionary Computation Conference*, pages 1177–1184. ACM, 2013.

[25] Salvador García, Daniel Molina, Manuel Lozano, and Francisco Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics*, 15:617–644, 2009.

[26] Ilya Loshchilov. CMA-ES with restarts for solving CEC 2013 benchmark problems. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 369–376. IEEE, 2013.

[27] Nikolaus Hansen and Raymond Ros. Benchmarking a weighted negative covariance matrix update on the BBOB-2010 noiseless testbed. In *Genetic and Evolutionary Computation Conference*, pages 1673–1680. ACM, 2010.

[28] Graheme A. Jastrebski and Dirk V. Arnold. Improving Evolution Strategies through Active Covariance Matrix Adaptation. In *IEEE Congress on Evolutionary Computation*, pages 2814–2821, 2006.

[29] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

[30] Ahmed Kheiri and Ed Keedwell. A hidden markov model approach to the problem of heuristic selection in hyper-heuristics with a case study in high school timetabling problems. *Evolutionary Computation*, 25(3):473–501, 2017.

[31] Ahmed Kheiri and Ed Keedwell. A sequence-based selection hyper-heuristic utilising a hidden Markov model. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, pages 417–424, New York, NY, USA, 2015. ACM.

[32] Ahmed Kheiri, Edward Keedwell, Michael J. Gibson, and Dragan Savic. Sequence analysis-based hyper-heuristics for water distribution network optimisation. *Procedia Engineering*, 119:1269–1277, 2015. Computing and Control for the Water Industry (CCWI2015) Sharing the best practice in water management.

[33] Agoston E Eiben, Emile HL Aarts, and Kees M Van Hee. Global convergence of genetic algorithms: A markov chain analysis. In *International Conference on Parallel Problem Solving from Nature*, pages 3–12. Springer, 1990.

Research highlights:

- Optimizing wind turbines positions.
- Use of evolutionary algorithm.
- Report on the 2015 wind farm layout opitmization competition.
- Compare new algorithms with state-of-the-art approaches.