On Learning to Generate Wind Farm Layouts

Dennis Wilson, Emmanuel Awa, Sylvain Cussat-Blanc, Kalyan Veeramachaneni, Una-May O'Reilly

On Learning to Generate Wind Farm Layouts

Dennis Wilson
CSAIL - MIT
32 Vassar Street
Cambridge, MA 02139, USA
dennisw@mit.edu

Emmanuel Awa
Brandeis University
415 South Street
Waltham, MA 02453, USA
eawa@brandeis.edu

Sylvain Cussat-Blanc
University of Toulouse
21 allée de Brienne
31015 Toulouse, France
cussat@irit.fr

Kalyan Veeramachaneni
CSAIL - MIT
32 Vassar Street
Cambridge, MA 02139, USA
kalyan@csail.mit.edu

Una-May O'Reilly
CSAIL - MIT
32 Vassar Street
Cambridge, MA 02139, USA
unamay@csail.mit.edu

ABSTRACT

Optimizing a wind farm layout is a very complex problem that involves many local and global constraints such as inter-turbine wind interference or terrain peculiarities. Existing methods are either inefficient or, when efficient, take days or weeks to execute. Solutions are contextually sensitive to the specific values of the problem variables; when one value is modified, the algorithm has to be re-run from scratch. This paper proposes the use of a developmental model to generate farm layouts. Controlled by a gene regulatory network, virtual cells have to populate a simulated environment that represents the wind farm. When the cells' behavior is learned, this approach has the advantage that it is re-usable in different contexts; the same initial cell is responsive to a variety of environments and the layout generation takes few minutes instead of days.

Keywords

Layout optimization; Developmental model; Gene regulatory network; Machine learning

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. The cost function is usually limited to minimizing inter-turbine wake interferences and thus maximizes 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.

Under these circumstances, we argue that the direct search approaches based on global optimization techniques are extremely inefficient. In such approaches, the environment is tessellated and an optimization algorithm, generally based on CMA-ES (Covariance Matrix Adaptation Evolution Strategy) or on a genetic algorithm, locally optimizes the turbines positions by slightly moving them. When the fundamental constraints or other aspects of the design problem are changed, the layout optimizer has to be re-run with a new starting point. When the approaches are population-based they rely on evaluating a population of layouts in each iteration causing the algorithm to sometimes run for days at a time. This latency creates a bottleneck in the design process. An algorithm that is efficient and can handle additional constraints during the design process is highly desired.

To start to address weaknesses of the current approaches, we propose a novel approach that uses a cell-based developmental model to "grow" wind farm layouts. Cells are a metaphor for turbines in a virtual layout. An evolutionary algorithm optimizes the cell's behavior with the aim to populate the environment so that the global network, the farm layout, maximizes the energy output or the earnings calculated by a cost-benefit (return on investment) simulation of the wind farm. The cells have to respect the local and global constraints and position themselves optimally in the environment. To do so, the cells sense the wind coming from various directions and decide what to do: divide, reorient their division plan, migrate, and more. This bio-inspired approach, commonly named the developmental model or morphogenetic engineering, has been used on complex problems since the end of the 1990's. Nowadays, developmental

models are commonly used to grow shapes or solve simple combinatorial functions [9, 2, 13, 21, 8, 5]. However, their applications to real-world problems are still to be explored. Producing modular robot morphologies [6] or the topology of optical fibers [16] are two examples of existing applications. After a learning process, usually based on a genetic algorithm that optimizes an encoding of the cell behavior, they have been shown to solve highly locally constrained problems and to scale up easily to harder problems without additional learning [4].

This paper is organized as follows. In §2, we present the background in wind farm layout optimization and the limitations of existing methods. Then, in §3, we present the wind farm evaluation function, the developmental approach that generates the layouts, and the gene regulatory network that controls the cells. §4 presents experiments that denote the properties of the evodevo model such as its scalability and its adaptativity to the environment. In §5 we conclude and consider what further aspects of the layout optimization problem can be addressed by out approach.

2. BACKGROUND AND LIMITATIONS

2.1 Background

Approaches to solving the wind turbine layout problem can be broadly sorted into three categories. In the first category, the approaches divide the site into a number of cells and use a discrete optimization algorithm that decides whether or not to place a turbine in the cell. A genetic algorithm optimizes the corresponding boolean matrix [17, 11, 12, 26, 10, 19, 27].

In the second approach, a continuous space search algorithm is used to move each turbine locally to identify the optimal placement. For example, a particle swarm optimization algorithm is used in [25, 3, 22]. A particle represents a turbine layout by a vector of $N(x, y)$ Cartesian coordinates, where N is the theoretical maximum number of turbines to place in the farm. A constraint-repairing algorithm keeps the layout coherent in regard to a secure area constraint.

Finally, a third set of approaches start with a grid layout and generate slight modifications on an existing layout by the use of CMA-ES to introduce a stochastic component [24]. Each decision variable, i.e. one of the coordinates of the turbines, is perturbed by adding a normally distributed random value. The standard deviation is also evolved using the members of the population that have survived a selection process based on their fitness. The algorithm learns the stochastic perturbations that will optimize the solution by learning the correlation among variables. This learning is performed by maintaining a matrix of covariances among the distributions controlling the perturbations of decision variables for every layout. This internal management allows the algorithm to drastically reduce the number of parameters involved in tuning.

2.2 Limitations

Considering the current needs and requirements of the layout design process, existing approaches have some fundamental drawbacks which motivate us to resort to a developmental model.

Representation issues: In the discrete version, a GA has very limited flexibility in turbine location as it is restricted to discrete cells. However, these cells can be skipped, providing a way to generate layouts with a variable number of turbines. For algorithms that search for locations in the continuous domain, the number of turbines is fixed giving limited flexibility for the cost function to affect turbine number. Additionally, proximity constraints are often violated, requiring sophisticated repair approaches.

Problem dimensionality: When the problem is solved via the discrete version, an x km by x km farm is divided into m^2 km cells giving the problem a binary dimensionality of $\frac{x^2}{m^2}$. On the other hand if the problem is solved via a continuous version the number of dimensions is $2 \times N$ where N is the number of turbines.

Reusability: One fundamental drawback of direct search approaches is that the algorithm has to be run every time there are changes in the problem space, or when additional constraints are added. Fully realistic layout evaluation tools such as OpenWind by AWS Truepower take so long that they are not practical to use in an iterative algorithm; an example optimization using this software to evaluate fitness would take approximately one week.

The objective of this work is to address the problems mentioned above by means of a developmental approach. A developmental model works differently in that, rather than generating a solution, it learns a parameterized function which, when executed a certain number of times, produces an entire solution. At each iteration, the function is sensitive to conditions affecting energy capture and cost of energy. As long as the evaluation model is the same, the function can be executed for a new farm, with different conditions, without having to re-learn.

3. LAYOUT EMBRYOGENESIS

3.1 Developmental model

The developmental model is based on the replication of virtual cells of size X meters by X meters that occupy a virtual environment. The developmental process starts with a single cell positioned in the middle of the environment. As shown on Figure 1, each cell senses the amount of energy it receives from eight different directions. These amounts are computed based on the wind speed distribution described as part of the farm specifications with the energy lost to wake effects, i.e. the interference of other turbines, subtracted. The inter-turbine interference model is described in §3.3.

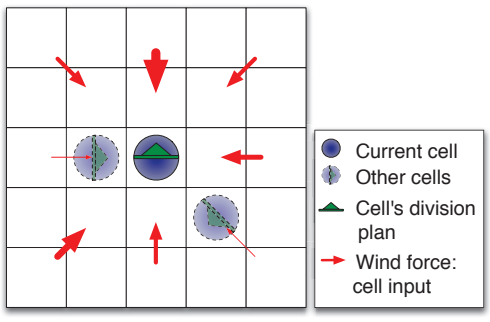

Figure 1: Cells act in a 2-D discrete environment. They decide the best action by the mean of 8 wind forces provided by the energy capture function.

With these eight different inputs, a cell has to decide which action to trigger at each step of the simulation. The possible actions for a cell are to *divide* in the direction defined by the cell's division plan, *reorient clockwise* the division plan 45°, *reorient counterclockwise* the division plan 45°, *wait*, and *kill itself* (apoptosis). This action list could be extended to give more degrees of freedom to the cells. However, these simple actions are sufficient such that a basic controller can easily be optimized and produce interesting results.

3.2 Site model

Cells are positioned in a 2-D flat terrain. This environment is represented with a discretized matrix. Each cell of the matrix represents an area of X meters by X meters. A wind distribution in 24 directions is given at the beginning of the simulation and is the same across the terrain.

3.3 Inter-turbine interference model

The simulation takes turbine wake interference into account using the following common energy capture model (for details, see [24])

$$\eta(X, Y, v, \beta(v)) \tag{1}$$

where X, Y are the coordinates for the turbine, v is the wind speed, and the function $\beta(v)$, known as a power curve, gives the power generated by a specific turbine for a given wind speed. Wind speed v however is a random variable with a Weibull distribution, $p_v(v, c, k)$, which is estimated from wind resource data. This distribution also changes as a function of direction, θ which varies from $0^0 - 360^0$, yielding a probability density function for different θ given by $p_v^\theta(v, c, k)$. Additionally, wind flows from a certain direction with some probability $P(\theta)$. These different pieces of information are inputs to the algorithm. Due to the random nature of wind velocity, the objective function evaluates the *expected* value of the energy capture for a given wind resource and turbine positions. For a single turbine, this value can be calculated using

$$E^i[\eta] = \int_\theta P(\theta) \int_v p_v^\theta(v, c_i, k, _i\, x_i, y_i, X, Y)\beta^i(v). \tag{2}$$

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. c_i and k_i are turbine specific resource parameters derived for the i^{th} turbine after wake calculations. For more details, refer to [15].

3.4 Cells' controller: GRN

As in nature, the cell's controller is a gene regulatory network (GRN). A GRN is a network of proteins that controls the behavior of the cells. In a living organism, a cell has several functions described in its genome. A gene regulatory network controls their expressions by the use of external signals collected from protein sensors localized on the membrane [7]. These signals activate or inhibit the transcription of the genes, which then determines the cell's behavior.

In our model, a similar network of proteins is optimized in order to generate the simulated cells' behaviors. The amounts of energy production sensed by the cells is translated to protein concentrations that feed the GRN. Every output protein chosen in the network is plugged to each possible cell action. When a cell has to act, it will choose the action with the highest output protein concentration. This kind of controller has been used in many developmental models of the literature [13, 8, 5] and to control virtual and real robots [18, 14].

We have based our regulatory network on Banzhaf's model [1]. It is designed to be as close as possible to a real gene regulatory network. It has been neither designed to be evolved nor to control any kind of agent. However, Nicolau used an evolution strategy to evolve the GRN to control a pole-balancing cart [18]. Though this experiment behaved consistently, the evolution of the GRN has been an issue. We have decided to modify the encoding of the regulatory network and its dynamics. In our model, a gene regulatory network is defined as a set of proteins. Each protein has the following properties:

- Its *identifier* (*id*) is coded as an integer between 0 and p. The upper value p of the domain can be changed in order to control the precision of the GRN. In Banzhaf's work, p is equivalent to the size of a site, which is 32 bits. We have kept the same precision by setting p to 32.
- Its *enhancer identifier* (*enh*) is coded as an integer between 0 and p. The enhancer identifier is used to calculate the enhancing matching factor between two proteins (see equation 3 hereafter).
- Its *inhibitor identifier* (*inh*) is coded as an integer between 0 and p. The inhibitor identifier is used to calculate the inhibiting matching factor between two proteins (see equation 3 hereafter).
- The *type* determines if the protein is an *input* protein, the concentration of which is given by the environment of the GRN and which regulates other proteins but is not regulated, an *output* protein, the concentration of which is used as output of the network and which is regulated but does not regulate other proteins, or a *regulatory* protein, an internal protein that regulates and is regulated by other proteins.

The dynamics of the GRN is specified as follows. First, the affinity of a protein a with another protein b is given by the enhancing factor u_{ab}^+ and the inhibiting u_{ab}^-:

$$u_{ab}^+ = p - |enh_a - id_b| \;\; ; \;\; u_{ab}^- = p - |inh_a - id_b| \tag{3}$$

where id_x is the identifier, enh_x is the enhancer identifier and inh_x is the inhibiting identifier of protein x. Figure 2 represents the GRN as a network of proteins (nodes) where the weights of the edges correspond to the affinity between the proteins.

The GRN's dynamics are calculated by comparing the proteins two by two using the enhancing and the inhibiting matching factors. For each protein in the network, the global enhancing value is given by the following equation:

$$g_i = \frac{1}{N} \sum_j^N c_j e^{\beta u_{ij}^+ - u_{max}^+} \;\; ; \;\; h_i = \frac{1}{N} \sum_j^N c_j e^{\beta u_{ij}^- - u_{max}^-} \tag{4}$$

where g_i (resp. h_i) is the enhancing (resp. inhibiting) value for a protein i, N is the number of proteins in the network, c_j is the concentration of protein j and u_{max}^+ (resp. u_{max}^-) is

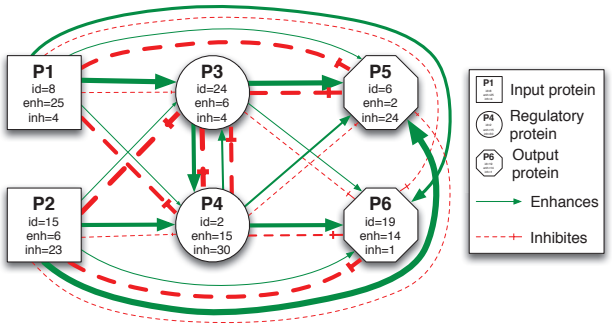

Figure 2: Graphical representation of a GRN: the nodes are the proteins and edges represents the enhancing and inhibiting affinity between two proteins. The bigger the edges, the closer the proteins.

the maximum enhancing (resp. inhibiting) matching factor observed. β is a control parameter described hereafter.

The final modification of protein i concentration is given by the following differential equation:

$$\frac{dc_i}{dt} = \frac{\delta(g_i - h_i)}{\Phi} \tag{5}$$

where Φ is a function that keeps the sum of all protein concentrations equal to 1.

β and δ are two constants that modify the dynamics of the network by setting up its speed of reaction. β affects the importance of the matching factor and δ affects the level of production of the protein in the differential equation. The lower both values, the smoother the regulation. Similarly, the higher the values, the more sudden the regulation.

Whereas the input proteins of a GRN corresponds to the current state of the environment, the output proteins select the best action to trigger. In the wind farm layout optimization problem, the inputs of the GRN of each cell correspond the wind energy coming from the 8 directions. They are provided by the evaluation model previously described. The output of the GRN corresponds to the cell actions: one output protein is assigned to each action; the highest concentration of the output determines which action the cell will do. If the action is not available (e.g. the cell wants to divide to a place where another cell is already living), the second highest protein's concentration is selected. The GRN is not reset before introducing the new input protein concentrations to maintain a persistent memory. The GRN is run for 25 steps with these new concentrations to reach a stable state before the decision is taken.

3.5 Optimization

The GRN can be easily encoded in a genome to be evolved by an evolutionary algorithm. The GRN's genome contains two independent chromosomes. The first one is defined as a variable length chromosome of indivisible proteins. Each protein is encoded within three integers between 0 and p for the three different identifiers. If an evolutionary algorithm has to evolve this chromosome, the variation operators have to be redefined. First, the *crossover* consists in exchanging subparts of two different networks. Because proteins are indivisible, the crossover points have to be chosen between two proteins. This ensures the integrity of each sub-network, and the local connectivity is maintained; only new links between the different sub-networks are created. The *mutation* can be applied in three equally probable ways: *mutating an existing protein* by randomly changing one of its three integers, *adding a new protein* randomly generated or *removing one protein* randomly chosen in the network.

Parameter	Value
Population size	500
Mutation rate	10%
Crossover rate	75%
Selection	3-player tournament with elitism
Minimum GRN size	8 input proteins + 5 output proteins + 15 regulatory proteins
Maximum GRN size	8 input proteins + 5 output proteins + 50 regulatory proteins

Table 1: Parameters of the genetic algorithm.

The coefficients β and δ presented in the dynamics model are encoded in a second independent chromosome which contains only these values. They are coded with double-precision floats in $[0.5; 2]$ (empirically chosen).

A genetic algorithm can then optimize the network of proteins and its dynamics. In this work, we have used a master-worker model distributed genetic algorithm in order to reduce the computational time. Table 1 presents the parameters used to evolve the gene regulatory network in this experience.

In order to globally evaluate the layout, we have designed the fitness function as a simplified economical simulation of the wind farm. A gain function computes the amount of money earned by a given layout over 20 years (average exploitation duration of a wind farm), or 175200 hours. Data used in computing revenue was taken from the US Department of Energy. The gain function is based on the energy capture, E, and the number of turbines, n, of the farm. Other constants are given in table 2.

$$C = E * R * 175200 - n * tc - \lfloor (n/T) \rfloor * sc - n * OM * 20 \tag{6}$$

This cost function is a better optimization criterion than maximizing the energy production because it is a more realistic economical model. However, it still does not address construction cost of the cable and the road networks, which are an important part of the global costs. Optimizing these facilities is a complex problem not addressed in this work.

Finally, the developmental process is completely deterministic: when triggered, the cell actions always give the same results. The GRN is also deterministic; for a given

Name	Description	Value
tc	Turbine Cost	\$750,000
sc	Substation cost	\$8,000,000
T	Turbine per substration	30
R	Revenue per kWh	\$0.08
OM	Yearly operation and maintenance costs	\$20,000

Table 2: Constant values of the fitness function.

l	θ^l	θ^{l+1}	k	c	$P(\theta)$	l	θ^l	θ^{l+1}	k	c	$P(\theta)$
0	0	15	2	7	0.0002	12	180	195	2	10	0.1839
1	15	30	2	5	0.008	13	195	210	2	8.5	0.1115
2	30	45	2	5	0.0227	14	210	225	2	8.5	0.0765
3	45	60	2	5	0.0242	15	225	240	2	6.5	0.008
4	60	75	2	5	0.0225	16	240	255	2	4.6	0.0051
5	75	90	2	4	0.0339	17	255	270	2	2.6	0.0019
6	90	105	2	5	0.0423	18	270	285	2	8	0.0012
7	105	120	2	6	0.029	19	285	300	2	5	0.001
8	120	135	2	7	0.0617	20	300	315	2	6.4	0.0017
9	135	150	2	7	0.0813	21	315	330	2	5.2	0.0031
10	150	165	2	7	0.0994	22	330	345	2	4.5	0.0097
11	165	180	2	9.5	0.1394	23	345	360	2	3.9	0.0317

Table 3: Realistic wind scenario parameters.

concentration of input proteins, the GRN will always provide the same output protein concentrations. Thus, running the developmental process only once is sufficient to calculate the fitness function of the resulting wind farm layout.

4. EXPERIMENTS

4.1 Cell learning

We use the wind distribution scenario and farm parameters from [15]. These problem parameters are standard in the literature and allows simple comparison between the different models. These parameter values provide the values for equations 1 and 2. They are presented in table 3.

First, the GRN is trained on a 14km by 7km 2-D flat field with the genetic algorithm and the mentioned wind farm evaluation model. The field is discretized into a 46x22 matrix of 310 meters by 310 meters vertices, which corresponds to the minimal security distance between two turbines to avoid inconsistent layouts. The genetic algorithm converges after 4 hours of learning on 128 CPUs, totaling 512 sequential hours. Due to the adaptable nature of the developmental approach, this learning has to be done only once. Therefore, we ignore this cost but reference the cost of layout growth when making the comparison with other methods.

Figure 3 graphically presents the developmental process that generates the layout. Here, only 4 of the 50 developmental steps are represented. A developmental strategy clearly emerges: the cells populate the environment following different development fronts, visible on the bottom left

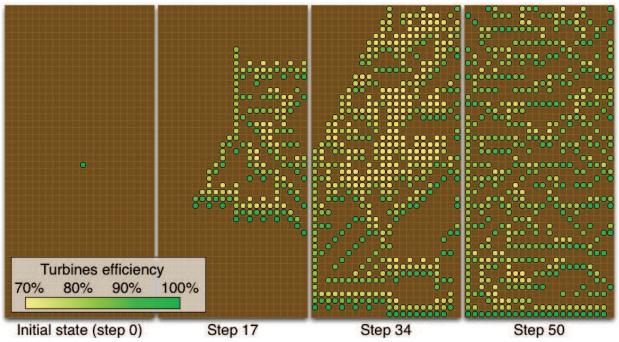

Figure 3: Growth of the best layout obtained with the developmental model.

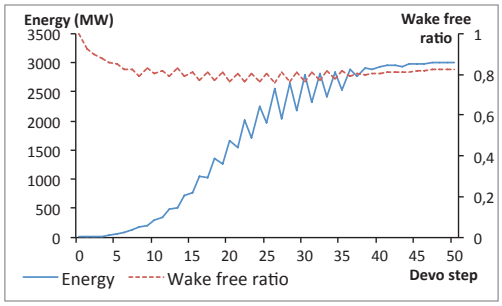

Figure 4: Variation of the energy output and the wake free ratio during the growth of the wind farm layout.

side of step 34. Moreover, three stages can be observed when these development fronts moves: first, the cells populate the local area as much as possible, the less efficient cells die, and finally the neighbor cells are optimized.

Figure 4 presents the energy output and the wake free ratio over the developmental process. The wake free ratio R_{wf} represents usage of the turbines; 100% means that all turbines are running at their theoretical maximum. $R_{wf} = \frac{E}{NE_{max}^{turb}}$, where E is the layout energy output, N is the number of turbines of the layout, and E_{max}^{turb} is the maximum theoretical energy output per turbine. Figure 4 clearly shows the exponential addition of turbines between step 0 and step 35. The energy output increases due to the cell proliferation. It is interesting to notice that the growth is interspersed with cell population extinction in order to improve the wake free ratio: inefficient cells die periodically to leave space to more efficient ones. This strategy allows a local optimization of the area. These extinctions disappear with the end of the cell proliferation.

The model is compared to the TDA 200k approach used in AWS OpenWind and in CMA-ES, both presented in [23]. Even if these approaches are grid-free, we choose them because they are the best methods currently used in the industry. However, both approaches use a fixed number of turbines while the developmental model also optimizes this parameter. To compare the developmental model with an algorithm that also optimizes the number of turbines, we have implemented a distributed master-slave genetic algorithm that optimizes the return on investment evaluation function we have designed. The genome of the genetic algorithm consists of a standard vector of bits. The GA is run for 200,000 layout evaluations like the CMA-ES and TDA approaches in [23]. Table 4 presents the results obtained by the developmental model.

	GRN	TDA 200k	CMA-ES	GA
Area (km^2)	14x7	14x7	14x7	14x7
Turbines	499	500	500	498
Energy (kW)	3.01e6	3.25e6	3.20e6	3.01e6
Wake free	**82.6%**	**88.8%**	**87.4%**	**82.7%**
Sequential time	**20sec**	**24h**	**324h**	**282h**

Table 4: Comparison of the developmental and the TDA 200k approaches on the same area size.

Area size (km^2)	Developmental approach					TDA 200k				
	Number of turbines	Energy output (kW)	Wake free ratio	Density of turbines	Computing time	Number of turbines	Energy output (kW)	Wake free ratio	Density of turbines	Computing time
10x6	340	2 018 410	**81.1%**	5.64	**10s**	300	1 971 000	**89.8%**	5.00	**11.3h**
12x6	400	2 369 780	**81.0%**	5.62	**13s**	400	2 584 000	**88.3%**	5.56	**17.0h**
14x7	499	3 014 420	**82.6%**	5.13	**20s**	500	3 249 000	**88.8%**	5.10	**24.5h**
20x10	1020	6 095 330	**81.7%**	5.10	**195s**	1000	6 449 000	**88.1%**	5.00	**75.0h**
-	Standard deviation		0.73%	0.30	-	Standard deviation		0.76%	0.27	-

Area size (km^2)	CMA-ES 200k					Genetic Algorithm				
	Number of turbines	Energy output (kW)	Wake free ratio	Density of turbines	Computing time	Number of turbines	Energy output (kW)	Wake free ratio	Density of turbines	Computing time
10x6	300	1 935 000	**88.2%**	5.00	**111h**	339	2 035 320	**82.1%**	5.63	**130h**
12x6	400	2 549 000	**87.1%**	5.56	**221h**	399	2 395 870	**82.1%**	5.60	**182h**
14x7	500	3 196 000	**87.4%**	5.10	**324h**	498	3 011 650	**82.7%**	5.12	**281h**
20x10	1000	6 298 000	**86.1%**	5.00	**327h**	1019	6 043 170	**81.1%**	5.10	**1247h**
-	Standard deviation		0.87%	0.27	-	Standard deviation		0.66%	0.29	-

Table 5: Comparison between the developmental and the TDA, CMA-ES, and GA approaches. The developmental model results are based on the controller obtained in the learning presented in section 4.1.

Because the developmental model and the genetic algorithm optimize both the number of turbines and the energy output, all four approaches can only be compared through the wake free ratio and the computation time. Actually, for the same area size, the developmental model and the genetic algorithm result in a comparable number of turbines (499 and 498 against 500 for TDA 200k and CMA-ES). The computation time is drastically reduced by the developmental model. Once the behavior is learned, only 20 seconds are needed to generate the wind farm layout instead of 24 hours required by the TDA 200k approach, 324 hours with CMA-ES and 282 hours with a genetic algorithm. Keeping in mind that manual iterations are necessary in the design process of a wind farm layout, the main interest of such a low computation cost is to keep the design interactive for humans. Designers can quickly evaluate the changes on the layout after the growth of a new farm that takes into account these changes.

The quality of the layout generated by the developmental model expressed by the wake free ratio is 6.2% lower in comparison to the TDA 200k. The same observation can be made with CMA-ES. However, the wake free ratio of the developmental model and the genetic algorithm are very comparable. Potential layout quality improvements will be discussed in the conclusion.

Out motivation in choosing our approach is context-sensitivity: the solution is built iteratively in a user-defined environment. To evaluate this property, we have performed two experiments using a learned GRN. In these experiments, the GRN is neither re-trained nor re-optimized. The first experiment consists of building wind farm layouts of different sizes and the second experiment introduces obstacles into the environment.

4.2 Scalability

In this experiment, we want to prove the scalability of our approach by introducing the same previously trained cell into three different environments. The sizes of these environments are the same as the environments tested in [23]: 10x6km^2, 12x6km^2 and 20x10km^2. The developmental approach is compared to TDA 200k and CMA-ES 200k taken from [23] and the genetic algorithm evolution presented previously.

Figure 5 presents the final layouts obtained with the developmental approach. On these layouts, we can note that the same pattern appears modulated to the environment sizes. For example, the bottom and the left cell lines are both present in all environments. They appear in the original layout (see figure 3) and on all layouts of different sizes. These patterns are important because they correspond to two good wind distribution values of our scenario. The most interesting point is that these patterns adapted to the field size.

Table 5 compares all four approaches with respect to number of turbines, energy output, turbine density, wake free ratio, and compute time.

The wake free ratio is comparable when the developmental approach is applied to various terrain sizes. The standard deviation of the wake free ratio is 0.73%. This standard deviation is comparable with other approaches. The same observation can be made for the turbine density in the environment (expressed in turbines per square kilometer). The

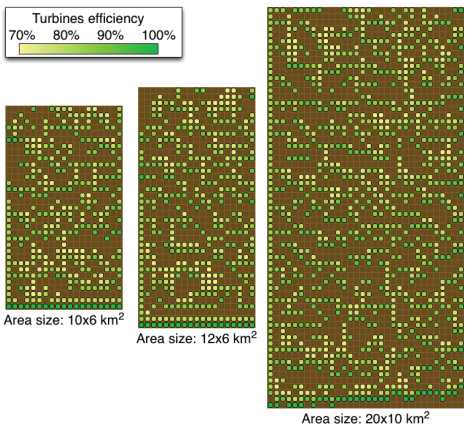

Figure 5: Reuse of the same initial cell and its GRN in environments of different sizes.

values stay stable with the terrain size variation with a standard deviation of 0.3, which is comparable to the standard deviation of the TDA 200k equal to 0.27 where the number of turbines is given by the user.

As in the context of learning the GRN, the wake free ratios are lower for the developmental model in comparison to the CMA-ES and TDA 200k and are comparable with the genetic algorithm approach. However, the computing time has been extremely reduced with the developmental approach. After the learning, the cell behavior can be exploited "as is" for different scenarios, whereas all other approaches have to be re-run to produce an optimal layout, requiring computing time in the order of days or weeks each time. Additionally, the developmental approach gives comparable results in small and large environments. This proves its capacity to scale up without further learning.

4.3 Avoiding natural obstacles

Another property of the developmental model is that it naturally takes into account unavailable positions in the environment. Here we again take the cells' GRN learned in §4.1 and we represent, in the environment, site features where turbines should not be placed, e.g. roads or water. We create a new vertex state that expresses unavailability; division into this cell area is prohibited.

To evaluate, we grow a new 14x7 km^2 layout. This time, the field contains a lake of 2.48x1.55 km^2 positioned near the bottom-right corner. Figure 6 presents the final layout obtained after 50 developmental steps. The energy output generated by this layout is 2.967MW with 499 turbines, which is only 1.58% less than the original layout. It has to be noted that the lake reduces the global area of 3.8km2 (3.9% of the whole field size). This layout is grown in 15 seconds which is comparable to the 20 seconds needed to develop the original layout. Finally, the wake free ratios are also comparable between both field: the layout with the lake has a 81.3% ratio and the original layout's one is 82.6%.

Visually, it is interesting to note that the developmental model was able to grow a front line on top of the lake; as the wind comes mainly from the bottom direction, these turbines are very efficient, as well as the turbines in the line produced on the bottom of the environment. However, the top left corner is not as populated as the original layout. The developmental model might have needed more developmental steps to populate that part of the field. Having a lake in the field might require additional growth which would extend the developmental process. The stop criteria of the development model is also a weakness of the current design because it is not context-sensitive; it has been empirically chosen throughout different runs. The model could be improved by the use of a stop criteria that depends of the wind farm layout optimization, such as a building cost threshold or average turbine efficiency.

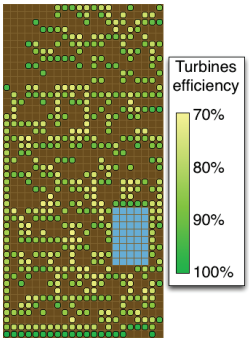

Figure 6: The developmental can easily avoid natural obstacles such as the lake represented by the bottom-right blue rectangle.

5. CONCLUSION AND FUTURE WORK

This paper presents a novel approach to generate wind farm layouts. It uses a bio-inspired approach based on cells that populate a virtual environment. The cells use inputs provided by an evaluation function that calculates the energy outputs of each turbine of the wind farm. To generate the layout, the cells can divide, reorient there division plan and die. A gene regulatory network controls the cells and is optimized by a genetic algorithm in order to give an appropriate behavior to the cells. The field is globally evaluated by a simplified economical simulation that calculates the return on investment of the wind farm over 20 years.

This approach has the advantages of scalability and context sensitivity; the cells act using local information and are evaluated by a global function. After an initial and single training, the cells can populate environments of various sizes. They can also take into account terrain constraints such as lakes, roads, and mountains. Other parameters that can be modified while still reaching an optimal solution with the trained GRN are yet to be explored. The wind distribution, turbine type, layout elevation topology, and revenue model are all important to the design process, and the effects of modification to these parameters on a trained GRN will be interesting. This approach is intended as the first step in such an exploration.

The main value of this approach is the drastic reduction in the computational effort; the same cell and its controller can be used in multiple environment without re-learning nor re-optimization. They always generate solutions with comparable qualities in term of turbine efficiency, highlighted here by the wake free ratio, and initial patterns created by the cells are adapted to the new environments. Thanks to the huge reduction of computing time implied by the use of our approach, the design loop is more interactive; human designers can modify on-the-fly the layout constraints and regenerate a new layout very quickly with the developmental model. This is not possible with other approaches with which days are necessary to optimize a new layout.

However, the solution quality is still lower than the solutions generated with other approaches of the literature. In our opinion, there are three ways to improve the quality of the developmental model to get closer to the TDA 200k and the CMA-ES results. First, the developmental model is based on a discrete environment. The cells and thus the turbines are positioned in the middle of a vertex of the grid. This enforces an alignment of turbines which strongly reduces the efficiency of the turbines. Secondly, the cells' controller, here based on a gene regulatory network, could be improved or even exchanged with another controller. For example, HyperNEAT [20] could more accurately select the cells' actions because the layout optimization problem has strong spatial properties. Other controllers such as neural network or classifier systems could also potentially yield better results. Finally, a hybrid between a local search al-

gorithm and the developmental model could improve layout quality; the developmental model could provide a good initial layout to be optimized by TDA or CMA-ES. Because the initial layout generated by the developmental model is better and more adapted to its environment than a randomly or uniformly generated layout, the local optimizers could be more efficient and converge more rapidly.

Acknowledgment: This work was performed using HPC resources from CALMIP (Grant 2013-P1319).

6. REFERENCES

[1] W. Banzhaf. Artificial regulatory networks and genetic programming. *Genetic Programming Theory and Practice*, pages 43–62, 2003.

[2] A. Chavoya and Y. Duthen. A cell pattern generation model based on an extended artificial regulatory network. *Biosystems*, 94(1-2):95–101, 2008.

[3] S. Chowdhury, J. Zhang, A. Messac, and L. Castillo. Unrestricted wind farm layout optimization (uwflo): Investigating key factors influencing the maximum power generation. *Renewable Energy*, 38(1):16–30, 2012.

[4] S. Cussat-Blanc, N. Bredeche, H. Luga, Y. Duthen, and M. Schoenauer. Artificial gene regulatory networks and spatial computation: A case study. In *Proceedings of the European Conference on Artificial Life (ECAL'11)*. MIT Press, Cambridge, MA, 2011.

[5] S. Cussat-Blanc, J. Pascalie, S. Mazac, H. Luga, and Y. Duthen. A synthesis of the Cell2Organ developmental model. *Morphogenetic Engineering*, 2012.

[6] S. Cussat-Blanc and J. Pollack. A cell-based developmental model to generate robot morphologies. In *Proceedings of the 2012 Genetic and evolutionary computation conference (GECCO)*, pages 2549–2556. ACM, 2007.

[7] E. H. Davidson. The regulatory genome: gene regulatory networks in development and evolution. *Academic Press*, 2006.

[8] R. Doursat. Facilitating evolutionary innovation by developmental modularity and variability. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 683–690. ACM, 2009.

[9] P. Eggenberger Hotz. Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes. *On Growth, Form and Computers*, page 302, 2003.

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

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

[12] H. 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.

[13] M. Joachimczak and B. Wróbel. Evo-devo in silico: a model of a gene network regulating multicellular development in 3d space with artificial physics. In *Proceedings of the 11th International Conference on Artificial Life*, pages 297–304. MIT Press, 2008.

[14] M. Joachimczak and B. Wróbel. Evolving Gene Regulatory Networks for Real Time Control of Foraging Behaviours. In *Proceedings of the 12th International Conference on Artificial Life*, 2010.

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

[16] S. Manos, M. C. Large, and L. Poladian. Evolutionary design of single-mode microstructured polymer optical fibres using an artificial embryogeny representation. In *Proceedings of the 2007 Genetic and evolutionary computation conference (GECCO)*, pages 2549–2556. ACM, 2007.

[17] G. Mosetti, C. 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.

[18] M. Nicolau, M. Schoenauer, and W. Banzhaf. Evolving genes to balance a pole. *Genetic Programming*, pages 196–207, 2010.

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

[20] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[21] G. Tufte. Metamorphosis and Artificial Development: An Abstract Approach to Functionality. In *10th European Conference on Artificial Life*. Springer, 2009.

[22] K. Veeramachaneni, M. Wagner, U. O'Reilly, and F. 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.

[23] M. Wagner, J. Day, and F. Neumann. A fast and effective local search algorithm for optimizing the placement of wind turbines. 2012.

[24] M. Wagner, K. Veeramachaneni, F. Neumann, and U. O'Reilly. Optimizing the layout of 1000 wind turbines. *European Wind Energy Association Annual Event*, pages 205–209, 2011.

[25] C. Wan, J. Wang, G. Yang, and X. Zhang. Optimal micro-siting of wind farms by particle swarm optimization. *Advances in Swarm Intelligence*, pages 198–205, 2010.

[26] F. Wang, D. Liu, and L. Zeng. Study on computational grids in placement of wind turbines using genetic algorithm. In *World Non-Grid-Connected Wind Power and Energy Conference*, pages 1–4. IEEE, 2009.

[27] C. Xu, Y. Yan, D. Liu, Y. Zheng, and C. Li. Optimization of wind farm micro sitting based on genetic algorithm. *Advanced Materials Research*, 347:3545–3550, 2012.