PAPER • **OPEN ACCESS**

Wind Farm Turbine Type and Placement Optimization

To cite this article: Peter Graf *et al* 2016 *J. Phys.: Conf. Ser.* **753** 062004

Wind Farm Turbine Type and Placement Optimization

Peter Graf (NREL), Katherine Dykes (NREL), George Scott (NREL), Jason Fields (NREL), Monte Lunacek (NREL), Julian Quick (NREL), Pierre-Elouan Rethore (DTU-Denmark)

Computational Science Center, National Renewable Energy Laboratory

E-mail: peter.graf@nrel.gov

Abstract. The layout of turbines in a wind farm is already a challenging nonlinear, non-convex, nonlinearly constrained continuous global optimization problem. Here we begin to address the *next* generation of wind farm optimization problems by adding the complexity that there is *more than one turbine type* to choose from. The optimization becomes a nonlinear constrained *mixed integer* problem, which is a very difficult class of problems to solve. This document briefly summarizes the algorithm and code we have developed, the code validation steps we have performed, and the initial results for multi-turbine type and placement optimization (TTP_OPT) we have run.

1. Introduction

Over the last several years, there has been significant growth in research on the design and optimization of wind plants [6]. Traditional wind farm optimization for placement of wind turbines involves a fixed turbine model and a description of the wind resources of a site (e.g., wind roses as a function of space). Even in the simplest scenario, wind plant layout design is already a challenging nonlinear, non-convex, continuous global simulation optimization problem. Here we add the complexity that the wind plant design may have more than one turbine type in the same project such that not only the turbine locations but their types are design variables of the optimization. Mathematically this is now a non-convex mixed integer nonlinear simulation optimization problem. There has been some effort in the past to apply MINLP solution methods to wind plant layout problems [4], but in that case the method was applied to optimization where the turbine locations themselves were discretized. In this study, we keep the turbine locations as continuous variables so that the discretization only comes in as a binary choice between wind turbines.

In developing a method for this problem we are immediately faced with a choice. If we reformulate or otherwise simplify our models in order to make them algebraic, quadratic programs, then algorithms (e.g. branch and bound) exist to optimize such models, and these algorithms provide rigorous bounds regarding how close we are to achieving optimality [3]. But this course takes us down a path this is both uncertain (how do we know our simplified model is accurate enough?) and time-consuming (we have to start from scratch and build an algebraic model of a wind farm, including wake effects).

Alternatively, we may implement a heuristic algorithm to approximately solve our simulation optimization problem. From a practical standpoint, such algorithms are often highly effective, and close examination of objective function values during the course of an optimization can often provide quite convincing evidence of near-convergence that is adequate for decision makers in that (1) the improvement over a baseline scenario is large and manifest and (2) the trajectory of the search is flattening, i.e. fewer and fewer better solutions are being discovered. For these reasons we have decided to pursue the heuristic algorithm approach.

2. Algorithm

The heuristic algorithm we have adopted for this project consists of a hybrid genetic algorithm. Genetic algorithms (GA) have proven to be particularly effective at so-called bit-string search problems in which the design variable is a vector of 0s and 1s [8]. The problem of turbine choice is just such a problem (the choice between more than 2 turbines can in principle be reduced to such a problem and will not be considered further here). Therefore a GA is a natural choice to handle the turbine choice (e.g., maximize annual energy production (AEP) by choosing turbine types at fixed locations) aspect of the search.

The turbine placement (local search (LS)) portion of the problem can be handled by any number of gradient-based optimization algorithms. For our studies we are using CONMIN because of its robustness in the face of the numerical noise of unconverged AEP calculations. Keep in mind, though, that these algorithms are simply local search. There is no guarantee of their finding the globally best solution without further augmentation.

Constraints: In the real world the turbines are subject to many constraints (electrical, mechanical, structural, financial, logistical, etc.). Our algorithm incorporates four constraints currently and is written to allow for additional constraints to easily be added by the user going forward:

(i) Lower bounds on distance between turbines (this is generally accomplished already by the optimizer combined with wake effects: wakes imply that it is *not* optimal to place all the turbines at the single point of highest wind)

(ii) Lower bounds on electrical cable length (minimal connected graph through turbines)

(iii) Lower bounds on distance from the edge of the site

(iv) Extreme wind constraint to achieve site-suitability of turbine placement (such that low-wind turbines are excluded from particularly high-wind-speed areas)

For the last constraint, we simply use the mean wind speed at a given location to estimate a 50 year maximum gust, then compare that to existing reference values (as given in IEC standards [7]) for the proposed turbine class to determine whether placing the turbine at the given site violates the extreme wind constraint. Note: In this document, the extreme wind constraint is only applied in section 5.

Given these algorithm parts, then, the question is how to best assemble the complete algorithm. We have adopted a hybrid GA strategy with some of the following ingredients:

- Our GA representation of a wind farm layout with a fixed number N of turbines consists of a list of turbine types t_i and their positions p_i. We can write this as either $x = \{(t_1, t_2, \cdots, t_N), (p_1, p_2, \cdots, p_N)\}$ or as $x = \{(t_1, p_1), (t_2, p_2), \cdots, (t_N, p_N)\}$ as convenient. In the former case the understanding is that p_i is the position of turbine t_i, whereas in the latter case that is presumably manifest.
- The initial GA population consists of layouts whose turbine types t_i and turbine locations p_i are both random.

- The GA operations of mutation and crossover act on both the turbine types $\{t_i\}$ and positions $\{p_i\}$. The local search acts only on the turbine positions $\{p_i\}$. The constraints are enforced whenever we move turbines or change turbine types.
- Every evaluation of the objective function $f(x)$ (e.g., f is AEP) involves some number of iterations of LS. The final positions after the local search are retained in x. That is, x evolves (via changes in p) during its evaluation. This is the aspect that makes this a hybrid genetic algorithm.
- Because changes to positions $\{p_i\}$ are retained across generations, the positions slowly evolve during the course of the GA evolution. This implies we need not run the LS to full convergence during the GA operation.

More generally, we observe that the choice of turbine type and choice of its placement are in some sense *separable*, i.e. almost independent (mathematically, a function $f(x, y)$ is separable if $f(x, y) = f_1(x)f_2(y)$ for some f_1, f_2). The primary drivers of AEP, the power curves, are such that for a fixed turbine type, every turbine–even low wind speed turbines–will favor locations with higher wind speed. The low wind turbine is just better than the high wind turbine if you do not have a choice about the wind (and it is low). Thus we can claim that for a fixed set of turbine types the optimal positions will be roughly the same. The converse is not true: for a fixed layout, the optimal turbines will depend on the wind speed there. This observation justifies our use of a semi-sequential algorithm in which positions are determined after turbine choice to some extent, but which requires iteration that provides feedback to ensure that turbine choice can be refined to reflect the positional degrees of freedom.

For example, a simplistic algorithm would simply be to use a GA to choose turbines for a fixed (say grid) layout of positions, and then run a single fully converged LS position search to finish the optimization. This would be highly efficient. However, there is then the potential that a fixed grid might dictate, say, t_i = A at p_i = Q when in fact near Q we should choose t_i = B to raise overall AEP. Our simplistic algorithm would miss this case. In the end the turbine choice and position are in fact not fully separable. Together these observations justify our use of a semi-separable approach in which short sections of LS are embedded in the GA, and we run a single fully converged LS at the end (in practice we find the local search is almost converged by the end of the GA, so the final full local search ends up being very short).

A further issue involves solving the *global* optimization problem robustly. For many realistic sites, the dependence of mean wind speed on location is non-convex (there are multiple peaks and valleys). Therefore it is likely that the globally optimal turbine placement might be highly irregular. It is therefore not likely to be the local solution starting from any particular fixed layout we can know ahead of time. This suggests there may be a further layer of the algorithm. A simple and often quite adequate approach to such situations is known as a multi-start (MS) algorithm. This simply means that every local search is started from not one but many starting points. This would be feasible in our case; inside the GA evaluations, we could solve the global problem by running a multi-start local search (MS LS) to convergence. This would presumably find the global best layout given the current choice of turbines. But it is highly inefficient.

In our case, though, we run a genetic algorithm on top of local search, and the GA randomizes not only the turbine types but their positions as well. That is, the MS approach is useful as a way to avoid a full-blown GA. But we have one anyway to solve the binary turbine choice problem. Allowing the positions to also undergo the randomization and mixing of the GA implicitly solves the global position problem at the same time as the turbine type problem.

The current TTP_OPT algorithm is summarized as follows:

(i) Input is choice of turbines, number of turbines, and map of wind speed and wind rose as a function of position in the wind farm.

(ii) Additional, algorithmic inputs include GA population size N_p, number of generations to run N_{GA}, number of LS iterations to run for each evaluation, and options related to parallelism (see below).

(iii) An initial population of wind farms is created, with different choices for turbine types and different baseline positions.

(iv) Now we run the N_{GA} GA iterations:

- Evaluate each member x of the population; recall this will invoke local positional search, and the new positions become x's positions in the GA population, so they can be carried forward.
- Based on fitness, select individuals for crossover and mutation. These operations inject modified versions of the best individuals back into the new population.

(v) After the GA completes, run a single, converged (more iterations) LS to arrive at the final optimized set of turbines and their positions.

Note that the objective function with which this document is primarily concerned is the optimization of AEP with constraints related to distance between turbines and tolerance to extreme wind gusts. Future work will also look at an overall cost of energy optimization, but that is beyond the current scope.

The TTP_OPT code we have assembled relies on several python and Fortran based codes, several of which are developed by ourselves and our collaborators. Among them are:

- OpenMDAO. An open-source multi-disciplinary, design, analysis and optimization software package from NASA Glenn Laboratories [9].
- FUSED-Wind. The Framework for Unified Systems Engineering and Design of Wind Plants is a wind-farm specific systems engineering framework develop jointly by NREL and DTU, using openMDAO [5].
- TOPFARM. This is a wind farm turbine layout optimization (local search only) software package built on FUSED-Wind developed by our collaborators at DTU Denmark [10].
- pyEvolve. This is a python-based genetic algorithm package. We have made several modifications to support hierarchical parallelism [1].
- pyOpt. This is a python-based (local) optimization package. We have made several modifications to support hierarchical parallelism [2].

3. Validation and Testing

The above discussion indicates that this is a complex problem and it is prudent to work our way up to cases of practical interest in a systematic fashion so that we believe the optimization tools are working as planned. Otherwise we cannot determine whether a non-intuitive final solution to a realistic problem is a brilliant discovery or the product of a bug in the code or a wrong assumption in the algorithm.

As a testbed we have run extensive simulations/optimization on a site we describe as a "Cliff". Specifically, this is a site where the wind is very high on half of the wind farm domain and low elsewhere. Importantly, the boundary of the cliff runs east-west, but the direction of the wind is a fixed 315 degrees (NW). This allows us to separate wind speed effects from wind direction effects. Separately, we have identified two test turbine models, one of which is most efficient at low wind (turbine "A"), the other most efficient at high wind (turbine "B"). Our first tests of the algorithms involve placements of "A" and "B" in the "Cliff" site.

The very first tests include the most simple operations such as verifying that the low wind turbine A really does produce higher AEP than the high wind turbine B on the low wind side of the cliff. From this we conclude that the scenario and the underlying solver are such that the GA should be expected to separate low from high wind turbines.

A simple test for the wake effects is to put two turbines in a steady (same speed, no cliff; call this the "Plain" scenario), unidirectional wind (in our case from the NW), constrain them to be a certain distance apart, and verify that the local optimizer can align them correctly. That is, if they are close enough that there might be wake effects, the two turbines should align such the line between them is perpendicular to the prevailing wind. This is a simple but important verification of the local solver and is exactly what we observe.

Next is a first proof-of-concept for the hybrid GA (i.e. including local search). For this, we optimize a 4 x 2 layout in the Cliff wind scenario (high wind on the north of the domain, low on the south, with fixed wind blowing from the northwest (in this case it was actually WNW)). Also, instead of fully randomizing the initial turbine positions, we start every member of the GA population with turbines located on the same baseline grid. Figure 1 shows the result, which is as expected. Here, purple is a high wind turbine, orange is a low wind turbine. The open circles show the initial grid layout, with a random distribution of high vs. low turbines. The closed circles show the final solution, where the GA has selected the appropriate turbines, and the LS has aligned them to be more orthogonal to the wind. (The reason they are not more orthogonal to the wind is most likely due to lack of convergence of both the local search and the underlying AEP calculations. Note that the high wind turbines are more orthogonal to the wind; the AEP is more sensitive to their alignment because the contribution from the high wind turbines is much larger.)

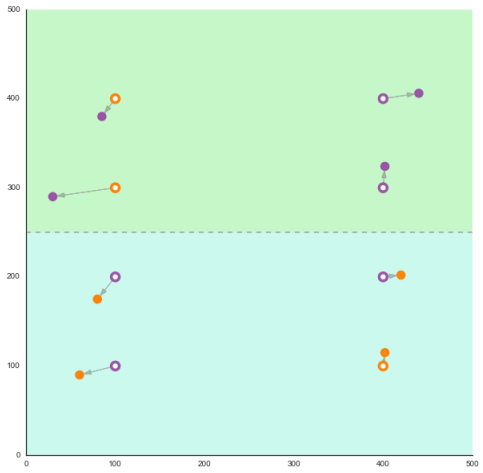

Figure 1. Simplest hybrid GA validation. The GA chooses the correct turbines, and the local search orients them optimally. The dotted line is the location of the "cliff" where the wind speeds change. The open circles show the initial grid layout, the closed circles show the final layout. Purple are high wind turbines and orange are low wind turbines. The wind is blowing from the west-northwest. As expected, the turbines align themselves roughly perpendicular to the prevailing wind to minimize losses due to turbine-turbine interaction.

Beyond the tests of the two optimization methods, local search and hybrid GA, the results of these preliminary studies are summarized in the appendix. In the first, an additional local search approach using a set of randomized conditions show that a multi-start approach is necessary to avoid local minima. In the second, the computational resources necessary for the hybrid GA method are determined through a test of the parallel scaling of the algorithm.

4. Results for a real site; fixed baseline positions, no extreme wind constraint

Here we describe initial tests for a real site. In this case we have mean wind speed data on a spatial grid of 25 meters over an approximately 7km x 5km site. We focus our studies on the northeast corner of the site, where wind speeds vary the most. The mean wind speed data is depicted as a 3D plot, with contours below, in Figure 2. In this case, there is complex terrain, but for the purposes of this study, the terrain is ignored (the site was chosen only as an exemplary wind resource data set with enough variability to demonstrate the optimization technique and not for use as a realistic design study).

There are two observations/predictions we can make about the wind resource for this site. First, the wind speeds are not particularly high, compared to those used in the validation tests above. Here all the speeds are closer to the low wind speed from our tests. That is, this is a class III site. So we expect that whatever the layout, the GA will pick the low wind speed turbine. (In this section we are using old model turbines of limited practical interest, so we will continue to label them A (low wind/class III) and B (high wind/class I). Second, we see that this is a complex wind resource landscape, so we expect the local optimization to result in local minima that are not necessarily global. Intuitively, we expect the local optimization to try to put the turbines on the peaks of wind velocity. Both these intuitions are born out by our initial studies.

Note that in this section, to separate the effects of the global versus local positional search, the initial turbine locations all start on the same fixed grid.

First, see Figure 3. This figure shows the results of a 16 turbine local optimization starting from a regular grid of locations. Indeed the turbines move uphill in wind speed to the hilltops of Figure 3, but this is by no means necessarily a globally optimal solution.

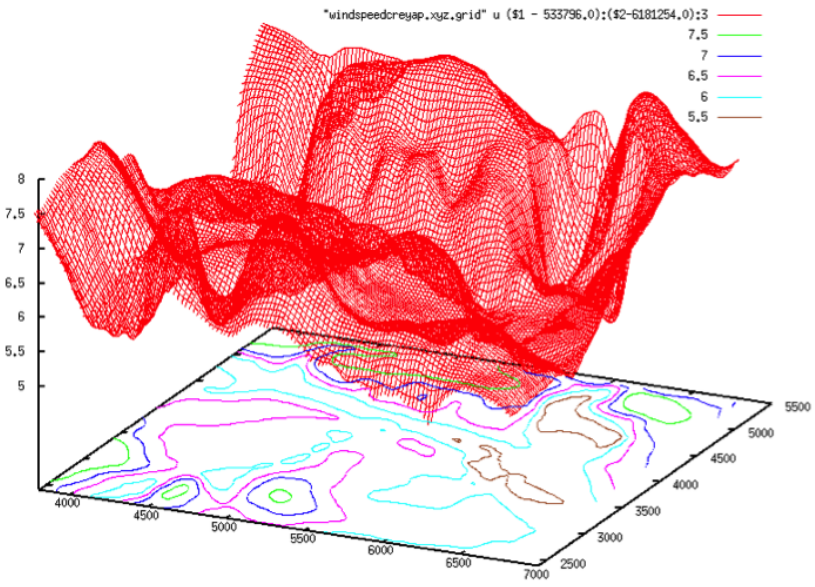

Figure 2. Surface and contour plot of the mean wind field data used in our initial study.

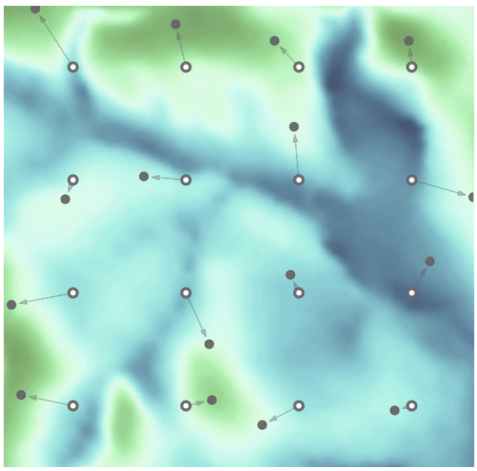

Figure 3. Local optimization of 16 turbines in the northeast corner of the study site. Open circles are initial positions, closed circles are final positions. The final positions indeed are generally on the "hilltops" of the wind speed landscape. There is certainly no guarantee that these are a global optimum. Note that there is a constraint in place that prevents the turbines from being within 360 m (4 * rotor diameter) of each other, which prevents them from always going to the top of the nearest hill. But the extreme wind constraint is not being enforced.

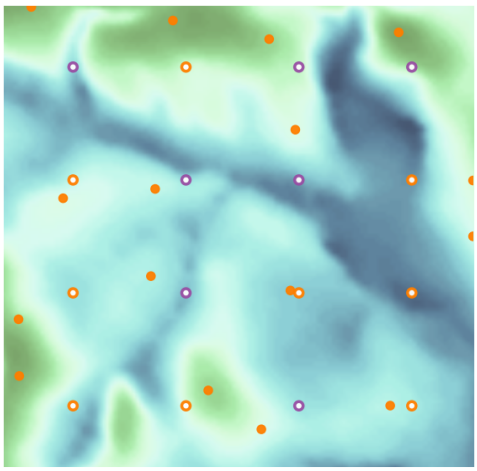

Figure 4. TTP_OPT hybrid GA turbine type and placement optimization for a real site. As above, the open circles are initial positions, the closed circles are final positions. Purple is a high wind turbine, orange is a low wind turbine. This site has wind suited to the low wind turbine. As expected, the GA chooses all low wind turbines. Note the general migration of the turbines toward higher wind speed.

Figure 4 shows the results of the hybrid GA optimization of the target site for a case of 16 (4x4) turbines. As expected, the low wind turbine is chosen by the GA. Also, as observed in the local search only, the turbines generally move toward "higher ground" (and because we are not yet enforcing the extreme wind constraint, they are free to do so). But, especially because we started all the layouts on the same fixed grid, systematic testing would likely show that the result is far from an optimal spatial layout.

5. Results for a real site; modern turbines, and including random position initialization and extreme wind constraint

Next we report on studies of distributing a mixture of realistic class I and class III turbines (for reference: turbine A is an IEC class III Alstom turbine with a rotor diameter of 122 m and a rated power of 2.7 MW and turbine B is a class I Alstom turbine with a rotor diameter of 100 m and a rated power of 3 MW). Here we have run the "full-GA", where the positions too are full fledged components of the genetic algorithm representation of the turbine, which means the turbine positions in each turbine layout are random, and they are subject to mutation and crossover just like the turbine types. Also, these results now enforce the extreme wind constraint. Non-intuitive mixtures and layouts of turbines emerge that increase energy up to 20% from the initial randomized conditions of turbine mixture and layout.

We focus here on runs with 16 and 64 turbines. For the 16 turbine case, we ran a GA with a population size 22 for 200 generations. To gain more confidence in the final result, and to understand the variability of the GA results, we ran the entire GA 50 times with different random seeds so that we end up with 50 different completely independent searches. The runs each used a single 24-core node on NREL's "peregrine" supercomputer, and each took 2 days for the 200 generations. Overall, with 50 trials, the study for the 16 turbine case took 57,000 CPU hours. For the 64 turbine case, only 80 generations were run (due to time and resource constraints) with 8 days and 240 cores per trial or over 1 million CPUs hours. Figures 5 and 6 show histograms of the maximal AEP values that were discovered as well as the GA "trajectory" for the best solution for each case.

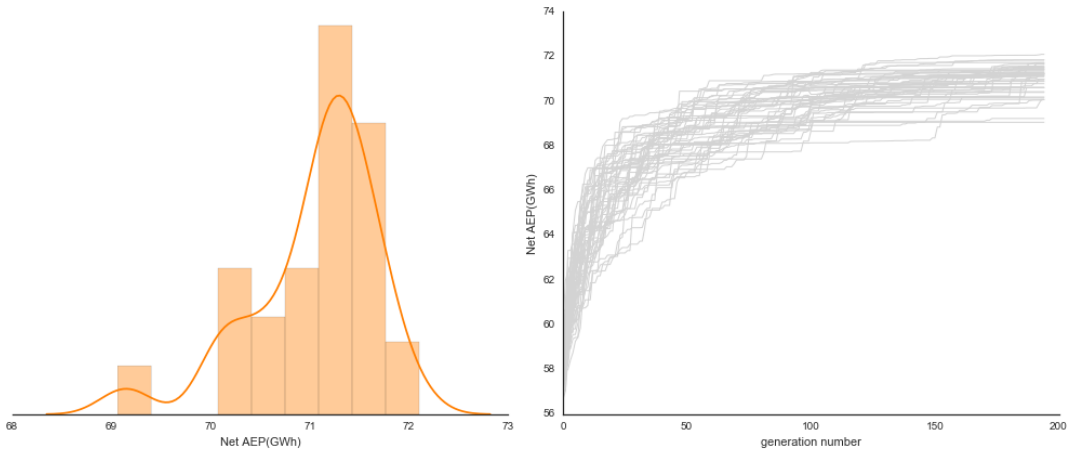

Figure 5. (left) Histogram of final AEP values for 50 independent GA runs for 16 turbine case. Clearly we benefit from multiple starts of the entire GA. (right) AEP as a function of generation for the GA run leading to the best solution for each of the 50 runs.

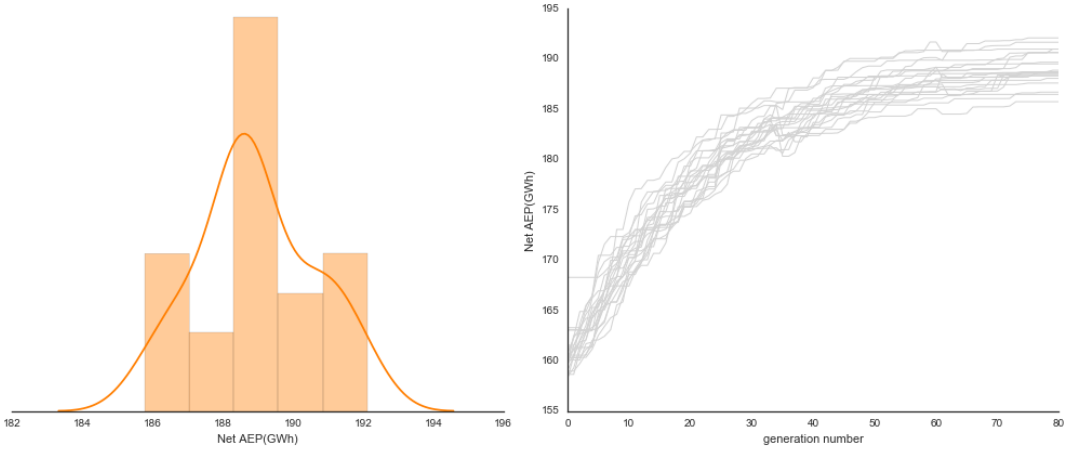

Figure 6. (left) Histogram of final AEP values for 50 independent GA runs for 64 turbine case. (right) AEP as a function of generation for the GA run leading to the best solution for each of the 50 runs.

We observe an approximately 20% increase (from 57 GWh for the random initial starting locations with 4 iterations of local search to 72 GWh for the final solution) in energy production due to the optimization (especially due to the *global* optimization). While there are some GA runs that are 4% below the best found, most are within a percent or two. Obviously, the more runs we can afford, the better the result will be, but we are near the point of diminishing returns for increased searching. For the 64 turbine case, with only 80 generations completed, the improvement in AEP is over 20% as well though there is less evidence of a fully converged optimization for the various trials. In both cases, further improvement could likely be obtained through additional starts and/or allowing the optimization to run for more generations.

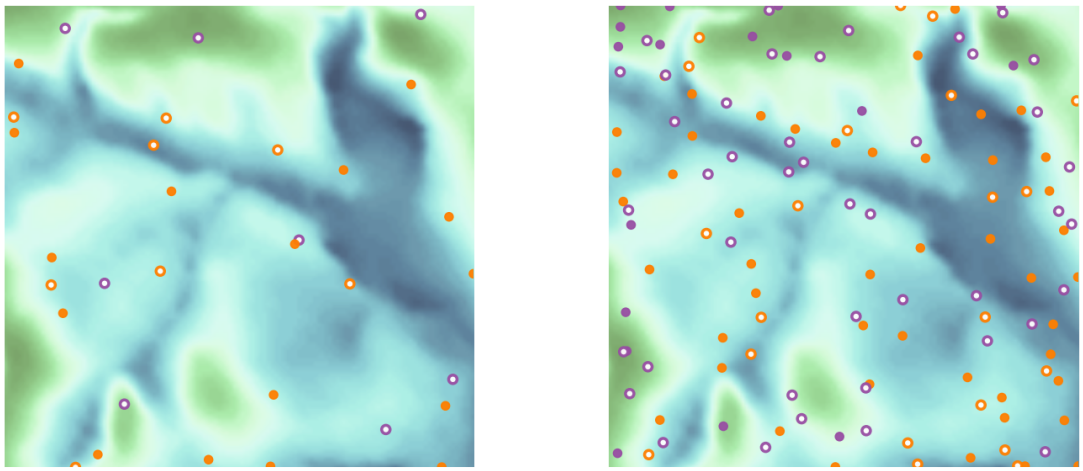

Figure 7. Best configurations found for the 16 turbine case (left) and 64 turbine case (right). Purple is class I (high wind), orange is class III (low wind). Open circles are representative starting positions, closed circles are final positions.

Figure 7 shows the best configurations found for the 16 and 64 turbine cases, as well as some representative initial configurations. Note the previous section showed that from an AEP standpoint the class III turbines should be selected. This is indeed true for the 16 turbine case. On the other hand, here we see several class I turbines appearing in the solution for the 64 turbine case. These are consequences of the extreme wind constraint combined with the need to keep minimum distances between wind turbines and also reduce the effects of wake losses. In the 16 turbine case, the turbines are able to spread out substantially (since there is no maximum distance constraint) and the wake losses for the final solution are relatively low and range from between 6-8%. In the 64 turbine case, on the other hand, turbine are indeed bumping up against the 4-D minimum spacing constraint and that coupled with wake losses are pushing turbines into the higher wind region. It would be possible to fit all turbines into the low wind speed regions and have all class III turbines, but the wake losses for such a scenario would become prohibitive. As it is, even utilizing these higher wind regions and class I turbines, the losses are on the order of 30-33%. The study here is meant to be illustrative of the technique given the simple models used and ignoring the actual site terrain, but it still illuminates the important trade-offs in a wind plant design between the available wind resource, site suitability and wake losses. For the last generation in the 64 turbine cases, all of the final designs had at least 17 class I turbines. There is a significant negative correlation between the number of class I turbines used at the site (ranging from 17 to 28) and the wake losses. The two best performing configurations had 17 and 19 class I turbines respectively and each had 191 GWh net AEP but with higher wake losses in the former case where more low wind turbines were crowded into the lower wind resource areas of the site. Even though the results are not fully converged, the results demonstrate that the final solution will have some percentage of high wind class I turbines to help reduce wake losses for an overall improved plant AEP.

Figure 8 are plots of the optimal solutions found for all 50 trials for the 16 and 64 turbine cases. For the 16 turbine case, we can see that there are still some class I turbines in the final solutions in the higher wind areas. These are actually associated with the lower performing trials and if run to full convergence, these turbines would likely disappear. Also, while there are many trials with similar energy production estimates, all the trials find final layouts where the class III turbines are clustered at the edge of the high wind regions that are prohibited by extreme wind constraints. For the 64 turbine case, the turbines are spread more uniformly across the entire wind resource space (excepting the lowest wind areas of both the low and higher wind resource regions). The spreading of the turbines more fully across the available design space is another indication of the need in the latter case to avoid wake losses which become prohibitive if the turbines are too closely spaced.

6. Results for a real site, modern turbines, and including random position initialization

Finally, we remove the extreme wind constraint and choose two turbines whose turbine power curves are similar enough that waking effects by themselves (rather than the exclusion for extreme winds) result in a situation where it may be more optimal to have two types of turbines in a single site. For this case, we use the same site as before but two different turbines: a Gamesa G114 2 MW (IEC class III) machine and a Vestas V112 3 MW (IEC class III) machine. In isolation, the Gamesa machine will produce more energy for lower wind resources than the Vestas turbine. Thus, in a wind plant environment where waking reduces inflow to wind turbines on the interior of the plant, the Gamesa has a chance to outperform the Vestas turbine. Figure 9 shows the results for a 16 turbine optimization maximizing net annual energy production where both turbine position and type are design variables along with the results where only the larger Vestas turbine is used in the optimization.

Firstly, we notice that the two best layouts of the optimizations in each case are very similar.

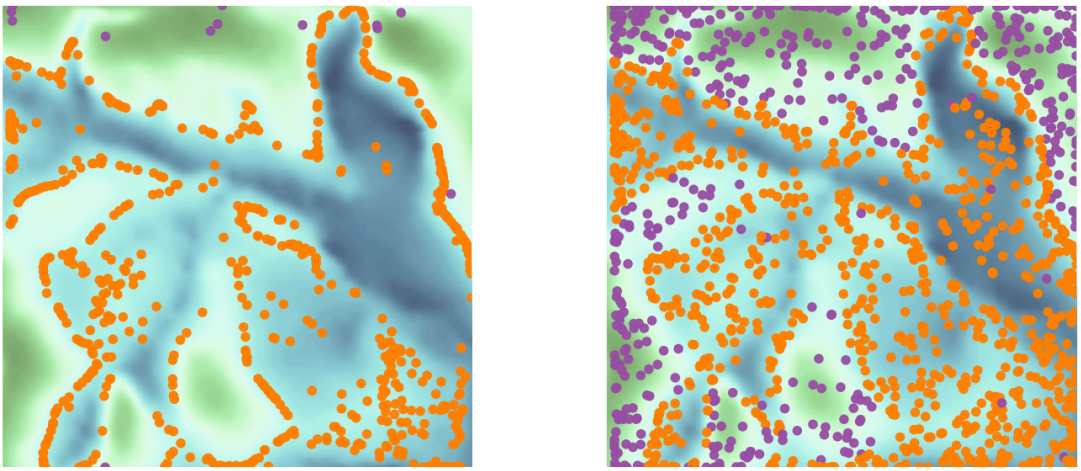

Figure 8. Combined results of all optimal configurations found for 16 and 64 turbine cases for all 50 trials in each simulation set. The 16 turbine case (left) shows the tendency towards selection of low-wind speed turbines appropriate for this lower wind speed site. The 64 turbine case (right) shows the tendency towards a mixture of turbine types.

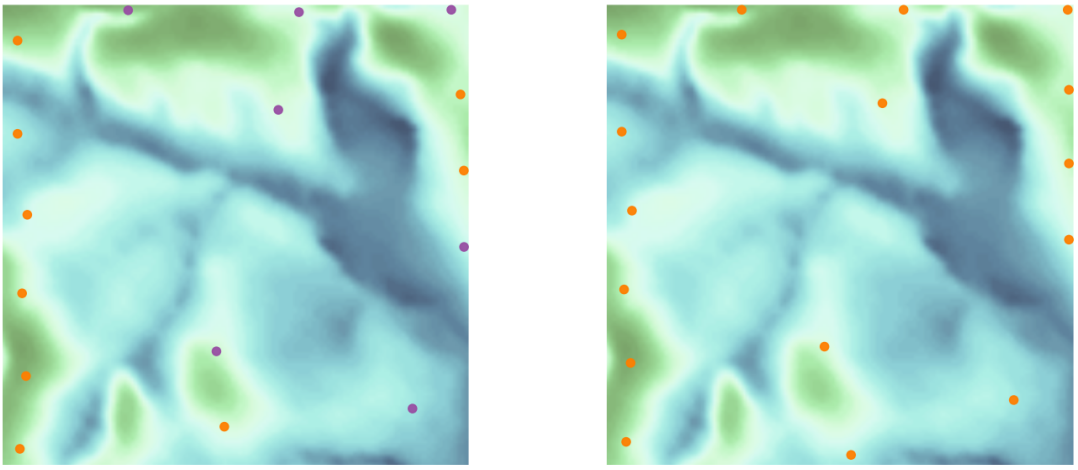

Figure 9. Results for the best optimization results for 16 turbine cases where both Gamesa and Vestas turbines are allowed (left) and only Vestas turbines are allowed (right). The Vestas turbines are orange in color and the Gamesa turbines are purple.

The solutions may not be fully converged and it is possible that the final solution for layout in both cases will converge to the same solution. Secondly, we notice in the mixed layout case that the optimum does have a number of Gamesa turbines in addition to Vestas turbines in the final solution. The prominent wind direction for the site is from the west so we see a large collection

of turbines along the western side and these are all Vestas turbines which perform better at higher wind speeds. For sites in the interior, Gamesa turbines, which perform better at lower wind speeds, are preferred. The waking effects of the plant are such that the aggregate effect is to lower the wind resource seen at the interior of the plant to the point where Gamesa turbines perform better. To make sure that the mixed layout is actually superior to the all-Vestas layout, the final layouts were run for both cases with the alternative turbine settings. Table 1 gives a summary of these results.

	Net AEP (GWh)	Gross AEP (GWh)	% Losses
All Vestas	64.4	67.8	5.0%
Mixed Types	64.5	67.4	4.3%
All Vestas / Layout Mixed Types	64.4	67.8	5.0%
Mixed Types / Layout All Vestas	64.3	67.4	4.5%

Table 1. Results of multi-turbine layout optimization.

The results show that the mixed turbine layout in the optimal layout configuration from its own optimization run produces the highest net AEP of any of the configurations. In addition, we note that the gross AEP of the All Vestas optimization is actually higher than the mixed turbine layout optimization in either layout, but the losses are significantly higher (5.0% versus 4.3%) so that the objective function net AEP is actually lower than for the mixed turbine layout. Thus, the mixed-turbine layout actually produces a better optimum for the site than a single-turbine design.

7. Conclusion

We have developed a hierarchically parallel algorithm to solve the mixed integer optimization of both turbine type and placement. We have presented several code validation examples, and we have presented results of applying the method to wind plant layout optimization with class I and class III turbines mixed in a single wind plant. This study is a first step towards including more complex decision making in the layout design process. On the one hand, intuitively it may be easy for a user to put low wind speed turbines in lower wind sites and high wind speed turbines in higher wind speed sites, but imagine trying to do such analysis manually for a large site where the potential number of plant configurations is huge and the optimal solution is non-obvious. In this study, this complexity was illustrated in the 64 turbine cases where the optimal combination of low and high wind speed turbines that would simulataenously maximize AEP (maximizing gross AEP, simultaneously minimizing wake losses) and meet site-suitability requirements (here represented via a maximum wind speed constraint). The final results show that there are potentially a large number of configurations with different combinations of turbines with similar AEP estimates. In a real design process, the collected final solutions from the optimization could be evaluated and compared according to a number of other criteria and potentially resulting in a final design that would be non-obvious and non-intuitive from a manual design approach. Future work will seek to add more realism to the optimization in terms of the wind resource, turbines and constraints. In addition, overall cost of energy will be considered as the design objective to incorporate key design considerations including balance of plant costs.

8. Acknowledgments

This work was supported by the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308 with the National Renewable Energy Laboratory (NREL). Funding for the work was provided by the DOE Office of Energy Efficiency and Renewable Energy, Wind and Water Power Technologies Office.

9. References

[1] Pyevolve. `https://pypi.python.org/pypi/Pyevolve`, 2016.
[2] pyOpt. `http://www.pyopt.org`, 2016.
[3] Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 5 2013.
[4] S. Donovan. An improved mixed integer programming model for wind farm layout optimisation. In *41st Annual Operational Research Society of New Zealnd Conference*, pages 143–152, Christchurch, New Zeland, 2006. Operational Research Soceity of New Zealand.
[5] DTU Wind Energy and NREL. FUSED-Wind. `https://github.com/FUSED-Wind`, 2016.
[6] J. Hebert-Acero, Probst O., P.E. Rethore, G. Larsen, and K. Castillo-Villar. A review of methodological approaches for the design and optimization of wind farms. *Energies*, 7(11):6930–7016, October 2014.
[7] IEC. 61400-1. wind turbines—part 1: Design requirements, 2005.
[8] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.
[9] NASA. OpenMDAO. `https://openmdao.org`, 2016.
[10] P.E. Rethore. TOPFARM. `https://github.com/DTUWindEnergy/TOPFARM/`, 2015.

10. Appendix A: Scaling and Multi-Start

This appendix describes additional tests for our approach including multi-start and scaling for parallelism.

In a first test, we have performed 24 separate local searches for a 10 x 10 turbine grid (no GA), where we have perturbed the initial conditions and run from 24 different starting layouts. Figure 10 shows the trajectories of these runs. This points at the necessity of ultimately incorporating a multi-start (or other globalization) approach into the final algorithm, as only a small number of runs achieved the best solution. As stated above, we accomplish this by allowing positions to be manipulated by the GA.

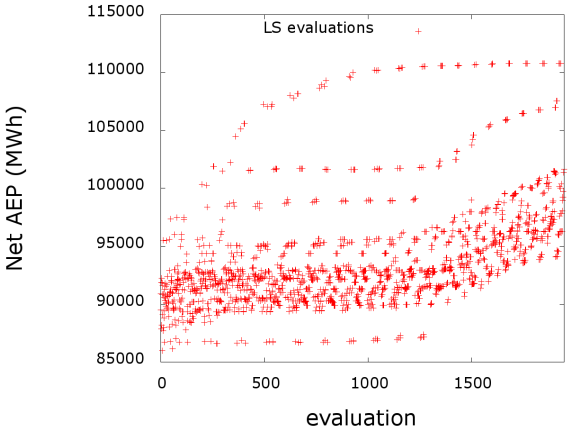

Figure 10. All evaluations from 24 local optimization from 24 different starting conditions. Note that (in the given number of iterations), only 4 out of 24 reached the best solution. This is a warning that we need multi-start or some other globalization approach to deal with the non-convexity of the local search phase of the turbine layout problem. But the GA builds this in.

The computations in the analysis are expensive computationally. Each AEP calculate requires about 1-10 seconds of serial computation (depending on number of turbines and fineness of wind speed and wind direction discretization). Each local search requires some number of iterations (e.g. between 2 and 20), each of which requires gradient calculations, which here are performed by finite differences, which requires an AEP calculation per degree of freedom (i.e., twice the number of turbines). Each of these LS operations is just the evaluation phases of a single member of the GA population. We find that for, say a 6x6 grid of turbines, we need roughly 20 iterations of a population of size 50. This description of the different levels of calculation required implies we need to evaluate AEP perhaps millions of times and suggests that we use parallel computing resources to accomplish our goal. But it also indicates how and where to parallelize.

Specifically, we are using what is known as hierarchical parallelism in order to subdivide the necessary computations in an inherently efficient fashion. First, recognizing that the evaluations of every member of a GA population are independent, we parallelize over these. Second, the computations of every component of the gradient are also independent, so we can parallelize over these. The code for this architecture runs efficiently on relatively large numbers of processors (we have run up to 1000 processors) and makes the above algorithm computationally tractable given access to NRELs supercomputer, Peregrine. We have been able to run the hybrid GA to completion for a 6x6 layout (36 turbine choices, 72 positional degrees of freedom) in a matter of hours with roughly 200 processors.

Figure 11 summarizes the parallel scaling of our code. First, we ran a 6x6 local search with

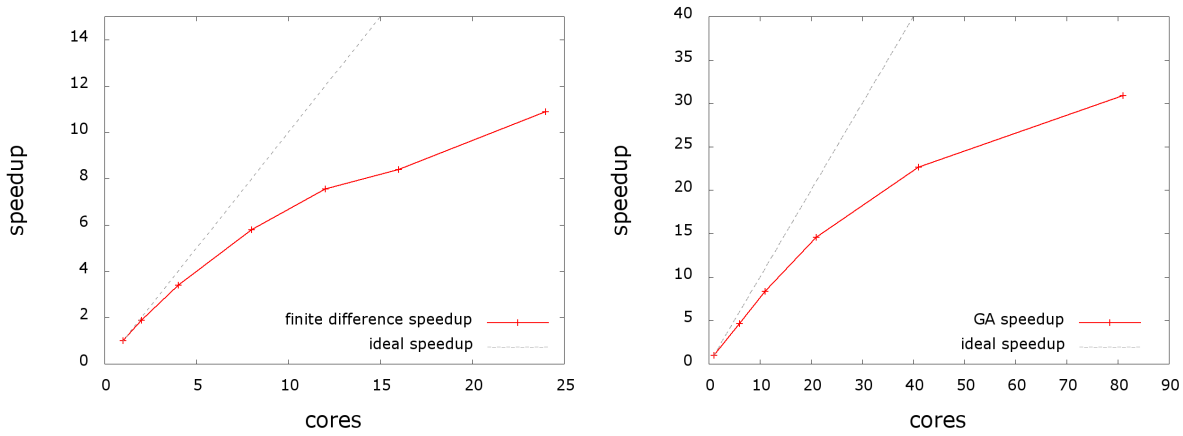

Figure 11. (left) Parallel scaling of local search phase, with 72 degrees of freedom. We get about a factor of 10 speedup with one 24 core node. (right) Parallel scaling of GA search with 100 turbines. 40 GA-evaluator groups results in about a factor of 20 speedup.

various numbers of processors. Next we ran a GA-only optimization of a 10x10 layout with varying numbers of processors. From these tests we may estimate the overall scaling we expect to observe, and estimate computational resources necessary to run a full target optimization. For example, here is a rough estimate: Our initial 10x10 GA-only test suggests we need no more than 5000 evaluations to converge, so let this be our target. We saw that a serial 10x10 local optimization took 12 hours. Elsewhere, observed scaling on a 6x6 case suggests we get a factor of 10 speedup on 24 cores, i.e., 24 cores could do one 10x10 local optimization per hour, i.e., one node-hour for each GA evaluation can be achieved, taking into account the observed rather than theoretical scaling of the finite difference gradient evaluations. I.e. 5000 node hours are needed if GA level is serial. Meanwhile, the 10x10 GA-only test reveals speedup over 20 at level of "40-GA evaluators", i.e., 40-way GA parallelism. This is about 50% efficiency compared to the scalar case. So our 5000 node-hours becomes 10000 actual node hours if we run at this level. So wall time will be $10000/40 = 250$ hours, on $40 * 24 \sim 1000$ cores. This is prohibitive given current computing resources.

What might save us? First, we probably only need 2500 GA evals. Next, as noted above, we probably only need a few CONMIN iterations in the GA phase, say a factor of 5 less (4 rather than 20 iterations). Also, the GA efficiency should go way up when the local search is "underneath" each evaluation, because unlike the test case these evaluations will take on the order of an hour rather than less than a second. So let's assume perfection at the GA level. Together this is 500 rather than 10,000 node hours (at 40x24 parallel layout). This is now $500/40 = 25$ hours on 1000 cores. This is feasible. In fact, the current implementation is more efficient than this (fewer LS iterations).