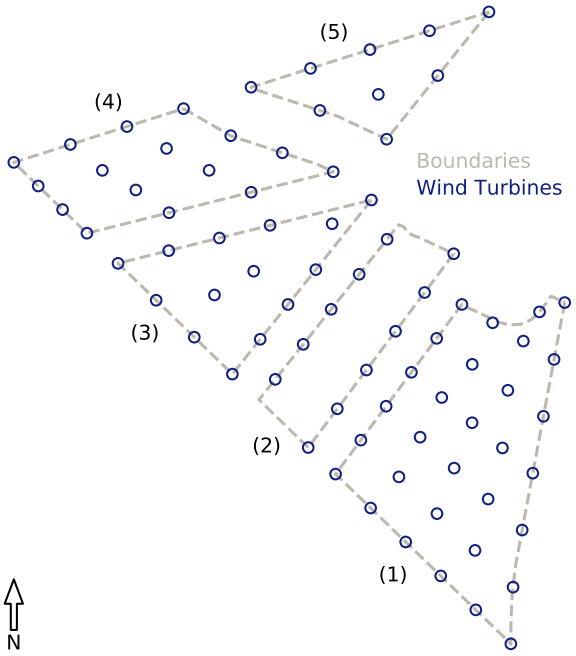

Figure 1. An overhead view of the wind farm used for the case study, including the provided example wind farm layout. Numbers in parentheses indicate region numbers. Wind turbine markers' diameters are the rotor diameters.

the-sum-of-the-squares method (Katic et al., 1986).

$$\left[\frac{\Delta V}{V_\infty}\right]_{total} = \sqrt{\sum_{k=1}^{N_T}\left[\frac{\Delta V_k}{V_\infty}\right]^2}, \tag{3}$$

where N_T is the number of wind turbines.

We chose AEP as the merit figure for the optimizations, with AEP defined as

$$AEP = \left[\frac{\text{hours}}{\text{day}}\right]\left[\frac{\text{days}}{\text{year}}\right]\sum_{i=1}^{N_D}\sum_{j=1}^{N_S} f_{ij}\sum_{k=1}^{N_T} P_{ijk}, \tag{4}$$

where the power of each turbine k for each wind direction i and speed j is represented by P_{ijk}; the probability of a given wind speed and wind direction combination is given by f_{ij}; N_D and N_S represent the number of wind directions and wind speeds respectively. The objective function can then be defined as

$$\begin{aligned} &\text{maximize} \quad AEP(x_i, y_i) \;\; i = 1...N_T \\ &\text{subject to} \quad S_{ij} \geq 2d \;\; i, j = 1...N_T \;\; i \neq j \\ &\qquad\qquad [x_i, y_i] \in \Omega \;\; i = 1...N_T, \end{aligned} \tag{5}$$

where S_{ij} represents the spacing between turbine i and turbine j and Ω is the set of all points in the defined boundary regions (see Fig. 1).

The objective function code was provided in the Python programming language, but some authors chose to re-implement the code in a different language. The wind resource, shown in Fig. 2, was divided into 360 different wind direction bins, and the wind speeds were assumed to follow a Weibull distribution, with 20 speed samples per wind direction. We increased the number of wind directions from the wind rose given in the original case study documents to make the problem more realistic. The wind speed probability distributions were unique in each direction. The complete wind rose definition is available in the supplemental data repository (see *Code and data availability* at the end of the manuscript). A more complete description of the case study prompt can be found in Baker et al. (2021).

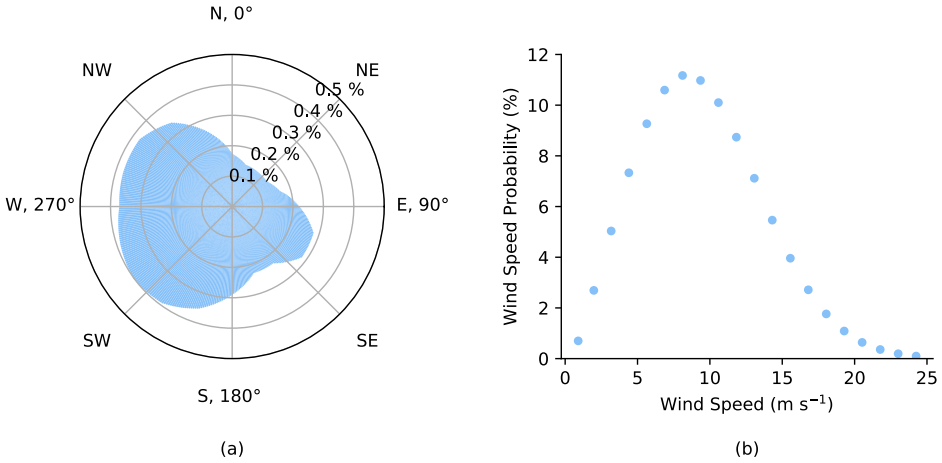

Figure 2. The full wind resource used for evaluating the final wind farm layouts. (a) The wind direction probability (360 bins). (b) A representative wind speed probability distribution (20 bins). The wind speed probability distributions were based on a Weibull distribution and represent the probability of wind at each speed in a given wind direction.

We compared the results of the optimization algorithms using a range of metrics in an attempt to capture some of the trade-offs between the algorithms. The simplest comparison was based on the objective merit figure, AEP. We also compared the layouts using wake loss, a common metric that indicates how much potential energy conversion was missed due to wake effects. We calculated wake loss, L_w, as

$$L_w = 1 - \frac{\text{AEP}}{\text{AEP}^*}, \tag{6}$$

where AEP^* represents the ideal AEP that would exist if all of the wind turbines were exposed to the freestream wind. Because most of the algorithms were run on different hardware, we do not compare run time. However, to provide some comparison of computational cost, we report the number of function calls run during the optimization. There are many trade-offs in the wind farm layout optimization problem that cannot be captured in a measurable way. We have also provided pro/con lists to help the reader understand some of the more qualitative comparisons of the algorithms.

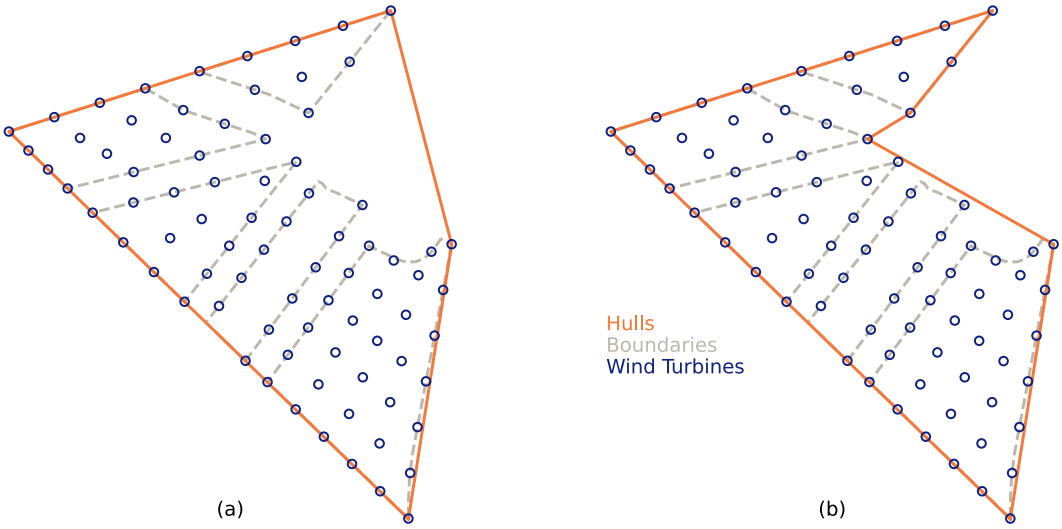

Figure 4. The convex and concave hulls used for the SNOPT+WEC method. (a) The convex hull as used in sub-optimization step 1. (b) The concave hull as used in sub-optimization step 2. The labels apply to both (a) and (b).

the WEC optimization series was complete, we ran a final optimization with the full wind rose (360 wind directions with 20 wind speeds in each direction as shown in Fig. 2) and $\xi = 1$. The sub-optimization step purpose, boundary, WEC factor values, convergence tolerance, and wind rose discretization for each sub-optimization in the series are shown in Table 1.

Table 1. Description of each sub-optimization for the SNOPT+WEC method.

	Sub-optimization								
	1	2	3	4	5	6	7	8	9
Purpose	Distribute	Allocate	WEC	WEC	WEC	WEC	WEC	WEC	Refine
Boundary type	Convex	Concave	Discrete	Discrete	Discrete	Discrete	Discrete	Discrete	Discrete
WEC factor	5.0	3.0	3.0	2.6	2.2	1.8	1.4	1.0	1.0
Tolerance	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}	2×10^{-5}
Wind rose	Reduced	Reduced	Reduced	Reduced	Reduced	Reduced	Reduced	Reduced	Full

Boundary and turbine spacing constraints were enforced using inequality constraints. The sign of the boundary constraint for each turbine represents whether the turbine is inside (negative) or outside (positive) of its assigned region. The sign is determined with a ray-casting algorithm based on the Jordan curve theorem (Press et al., 2007). For each turbine, a vertical ray is drawn and the number of intersections with its region's boundaries is counted. If the number of intersections is odd, then the turbine is within the boundary; if it is even, then the turbine is outside the boundary. The magnitude of a turbine's constraint value is equal to the distance from the turbine to the nearest point on the boundary of its assigned region. Thus, the larger the

2.2 Discrete exploration-based optimization (DEBO)

Discrete exploration-based optimization (DEBO) is a new algorithm developed specifically for this case study. DEBO aims to overcome two of the major difficulties of wind farm layout optimization. First, the two-dimensional domain where turbines can be placed, $\Omega \in \mathbb{R}^2$, consists of unconnected regions. This feature makes the overall layout optimization problem both continuous and discontinuous. The problem is discontinuous because the optimization algorithm has to find the best way to divide the turbines among the different regions of Ω. The problem also has continuous characteristics because once the turbine division is made, the turbines may be placed continuously within their respective regions to optimize the AEP. Second, some of the regions are not convex, and the function to be minimized has many local optima throughout the search space. This last feature suggests that we should not rely solely on local methods because local methods alone may converge to poor local optima. Taking into account the combination of discontinuous and continuous characteristics, as well as the large number of local optima present in the wind farm layout optimization problem, we have developed a purely discrete exploration-based algorithm (DEBO) that includes both a greedy initialization procedure and a local-search refinement process.

1. **Greedy initialization**: The greedy procedure sequentially places the N turbines in an a priori defined domain where the wake losses can be minimized. This domain includes the boundaries of the different admissible domain. When the right number of turbines has been placed, the initialization stops.

2. **Local search**: The local-search method sequentially, and in random order, places each turbine in its discrete neighborhood until the AEP converges. The local-search method is inspired by classical stochastic gradient methods (Wasan, 1969)[2]. A stochastic gradient step consists of modifying only one randomly chosen coordinate of the parameters while the other coordinates are fixed. The DEBO method relies on the same principle; one, and only one, randomly chosen turbine is moved at each iteration. However, the DEBO method differs from classical stochastic gradient methods since DEBO is not gradient-based but relies on a discrete exploration region with a varying radius. The radius of the discrete exploration method is reduced each time the layout reaches a local minimum, i.e., when no discrete turbine displacement can improve the AEP. The DEBO method's local search part also differs from the random search algorithm presented in Feng and Shen (2015). In Feng and Shen (2015), each turbine is moved randomly while it provides an increase of AEP, then another turbine is randomly selected and moved randomly. This differs from DEBO because each turbine displacement is not required to belong to a particular neighborhood. Therefore, unlike the DEBO method, the random search algorithm presented in Feng and Shen (2015) has no guarantee to eventually stop. In addition, the presented version of the DEBO algorithm is highly parallelizable.

2.2.1 Formulation of the problem

We first define the layout of a wind farm $\mathbf{F}$ to be a sequence of turbine coordinates, (x, y), such that

$$\mathbf{F} = \langle (x_1, y_1), \ldots, (x_{N_{\max}}, y_{N_{\max}}) \rangle, \tag{8}$$

[2]The stochastic gradient mentioned here is not the same as the stochastic gradient used in the context of deep learning, although these methods do have the same name.

2. It does not require discretization of the design space.

3. Its simplest variant can be used for any wake model (including computational fluid dynamics models) and the other variants can be used whenever wake effects can be attributed to individual waking turbines (such as with typical engineering wake models).

4. The concept behind it is flexible in the following ways:

   (a) The steps moving turbines used in the pseudo-gradient-based approach can also be used as steps in meta-heuristic approaches, replacing the typically more random steps used there.

   (b) Appropriate pseudo-gradients can be formulated for other concerns and aspects, such as the impact of bathymetry on substructure cost and cable layout.

5. An open-source implementation exists with a Python interface and efficient vectorized code.

Cons

1. The approach is prone to getting trapped in local optima. Therefore, it is suggested to use a multi-start approach or combine it with techniques such as wake expansion continuation (e.g., Sect. 2.1).

2. The heuristic nature of the approach "throws away" information available to true gradient-based approaches. Therefore we generally expect that, starting from an identical starting layout, gradient-based approaches will find better optimized layouts. (PG optimization is capable of investigating more starting layouts for equal computational cost.)

3. While the adaptive nature of the algorithm lessens the need for semi-manual parameter tuning (initial step size and step multipliers), the quality of the optimized layouts can be substantially affected by parameter selection.

2.8 Discrete perturbation algorithm (DPA)

The discrete perturbation algorithm (DPA) is a simple but effective gradient-free optimization method used in industry. The algorithm is based on a discretization of the design space with turbine locations tested by both directed and random perturbations from existing turbine locations. The algorithm proceeds as follows:

1. Determine the smallest bounding rectangle that can contain all possible locations of the turbines (i.e., a rectangle containing the site boundary) and any wind resource information.

2. Create a grid at a user-defined resolution, usually between 10 m and 50 m.

3. Determine whether each point in the grid is a legal turbine position (i.e., is it inside the site boundary and away from houses, roads, etc. as specified in the geographic information system layers). If the point is legal, add it to a 1D list of legal turbine positions for the current turbine layout.

4. Using the wind resource information, order the list from highest wind speed to lowest. This creates a continuous 1D sorted list of turbine positions for the optimizer to work with rather than a 2D, fragmented, unsorted search space. In the unlikely case that the wind resource is uniform across the wind farm, this sorting step will not do anything. The purpose

of sorting by wind resource is merely to avoid turbines jumping to positions which are significantly worse than their current position in terms of free-stream wind minus wake effects.

5. Repeat steps 1 to 4 for each turbine layout so that there is a sorted list of legal positions for each turbine layout.

6. Make sure the starting layout is legal. If it is not legal then make a legal layout from scratch (one way is to loop through the sorted list going from windiest to least windy while respecting the various constraints listed in step 8—this means that the starting layout is already as windy as it can be but needs to be spaced out to find the optimal trade-off between wakes and resource).

7. Test starting layout (for energy including wakes and all other losses) and keep a record of the net energy per turbine.

8. Perturb the layout so that each turbine has a new legal position that respects the geographic information system constraints as well as noise, visual, shadow flicker, and inter-turbine spacing constraints (circular, elliptical, or sector-wise as defined by the user). Two types of perturbation are available:

   (a) The priority is to jump to a better location from our sorted list of legal positions. However, we only jump to places that are higher in the list or whose wind speed is greater than the turbine's current waked wind speed plus some small tolerance (a different but similar heuristic is used for LCOE but from now on we will just talk about energy optimization). We make a few dozen attempts to jump to a new location, but if that is not working out (due to spacing, noise, etc.) then we try perturbation approach 8(b).

   (b) Random perturbations of the order of the grid spacing used to sample legal positions. The aim of this is to allow the turbines to fine-tune once they find there are no big jumps to be made. However, if a position opens up, there is no reason why this turbine cannot go back to making big jumps. Sometimes there is another turbine already in the way for this iteration. If a successful random perturbation was made in the last step, then the same vector is used for this one to allow momentum in moving to a better location.

9. Test the perturbed layout. If the net energy is higher, then accept the perturbed layout as the new starting layout and go to step 8 (this almost never happens but can happen right at the start of an optimization).

10. Look at the net energy results for each turbine. If the net energy given by the perturbed position is lower, then move this turbine back to its unperturbed position. If all turbines get moved back, then go to step 8.

11. Test the layout again. It is possible that turbine 1 may have only appeared to be doing better because turbine 2 had been perturbed to a worse location. With the return of turbine 2, the last move of turbine 1 may no longer be a good move.

12. Loop through steps 9 to 11 a maximum of three times before giving up and going to step 8.

13. Repeat steps 8 to 11 for as long as patience or time allow; the algorithm has no good stopping condition and tends to continue improving indefinitely.

finding optimal layouts. Continuous formulations provide more options, but suffer from having potentially too many options to explore effectively.

Because only one method was purely gradient-based, it is hard to draw many conclusions about the trade-offs between gradient-free and gradient-based algorithms. The gradient-based method, SNOPT+WEC, ranked third overall in terms of AEP and wake loss. SNOPT+WEC used relatively fewer function calls, and had low CPU time. This seems to indicate that at least this gradient-based method can achieve comparable results in less time than many gradient-free methods on the wind farm layout optimization problem. More work could still be done to reduce the computational cost and further improve exploration.

Few clear trends were shown between turbine-to-region allocation and layout quality. Three of the four allocation method types showed up in the top four layouts. However, because all methods moved turbines between regions, especially adding many turbines to region 5, we can say that allocating the right number of turbines to each discrete region in the farm is critical to finding the best layouts. Optimization methods that used penalty methods for turbine-to-region allocation did not perform as well as the those using the other allocation methods. The best layout by AEP was found using a sequential allocation method.

While it is important to weigh the quality of results and computational cost of each algorithm, the developer cost and level of expertise are also important to balance. Off-the-shelf algorithms, such as GPS and CMA-ES, may be well documented, but may not perform as well as newer methods not yet available in standard code packages. Commercial code implementations, such as SNOPT and GPS, require a license, but they are well documented. Open-source-only methods may take a little more effort to run but may also have more recent algorithm developments. Newly developed algorithms, such as DEBO, WEC, ADREMOG, and PG may be available through research codes or may need to be reimplemented. Implementations of industry methods like DPA may not be available, but could be approximated, which may require substantial effort for development and fine-tuning. Also important to consider are the additional efforts involved for various peripherals such as discretizing the design space, altering the wake model to accommodate algorithmic differentiation or WEC, obtaining gradients, preparing and running a multi-start optimization, reducing the wind resource, defining initial layouts, or setting up a turbine-to-region allocation method. When selecting which algorithm to use, carefully weigh the requirements and available resources in order to select the most suitable algorithm for the given situation.

While Herbert-Acero et al. (2014) refer to the no-free-lunch theorem as "for all possible performance measures, no algorithm is better than another when its performance is averaged over all possible optimization problems" (see Wolpert and Macready (1997) for more details on no-free-lunch theorems), Herbert-Acero et al. (2014) also point out that individual algorithms may be effectively tuned to perform well on specific optimization problems. In this work we found that none of the algorithms significantly outperformed the others and attribute this similar performance, in part, to the tuning of each algorithm to this problem by individual experts, which is one of the key differences between this and most of the previous comparative studies of optimization algorithms applied to wind farm layout optimization problems. The similarity may also be partly due to the problem being somewhat flat with many similar local optima. However, the DEBO algorithm was developed specifically for this case study and it did find the layout with the highest AEP. It will be interesting to see how the DEBO method performs in future comparisons and other problems.