Proximity search heuristics for wind farm optimal layout

Fischetti, Martina; Monaci, Michele

Published in:
Journal of Heuristics

Link to article, DOI:
10.1007/s10732-015-9283-4

Publication date:
2016

Document Version
Peer reviewed version

Noname manuscript No.
(will be inserted by the editor)

Proximity search heuristics for wind farm optimal layout

Martina Fischetti · Michele Monaci

Received: date / Accepted: date

Abstract A heuristic framework for turbine layout optimization in a wind farm is proposed that combines ad-hoc heuristics and Mixed-Integer Linear Programming. In our framework, large-scale Mixed-Integer Programming models are used to iteratively refine the current best solution according to the recently-proposed *proximity search* paradigm. Computational results on very large scale instances involving up to 20,000 potential turbine sites prove the practical viability of the overall approach.

Keywords Wind Farm Optimization, Heuristics, Mixed Integer Linear Programming

1 Introduction

Green energy became a topic of great interest in recent years. Indeed, environmental sustainability asks for a considerable reduction in the use of fossil fuels, that are pollutant and unsustainable. As a consequence, ambitious plans have been proposed for green energy production, including wind energy. The *wind farm layout optimization problem* consists in finding an optimal allocation of turbines in a given site, to maximize the power output. This strategic problem is extremely hard in practice, both for the sizes of the instances in real applications and for the presence of several nonlinearities to be taken into account. A typical nonlinear feature of this problem is the interaction among turbines, also known as *wake effect*. The wake effect is the interference phenomenon

M. Fischetti
DTU Copenhagen and Vattenfall BU Renewables, Oldenborggade 25-31, 7000 Fredericia, Denmark
E-mail: martina.fischetti@vattenfall.com

M. Monaci
DEI, University of Padova, Via Gradenigo 6/A, 35131 Padova, Italy
E-mail: michele.monaci@unipd.it

for which, if two turbines are located one close to another, the upwind one creates a shadow on the one behind. This is of great importance in the design of the layout since it results into a loss of power production for the turbine downstream, that is also subject to a possibly strong turbulence.

It is estimated in [2] that in large offshore wind farms, the average power loss due to turbine wakes is around 10-20% of the total energy production. It is then obvious that power production can increase significantly if the farm layout is designed so as to reduce the effect of turbine wake as much as possible.

Figure 1 illustrates the wind farm problem corresponding to a 3,000 $\times$ 3,000 (m) offshore area where turbines can be installed. The small circles identify the points where a turbine can potentially be built (*sites*), while filled circles refer to the currently built turbines. Interference due to the built turbines are represented in the background of the figure, and refer to the average interference over 500 macro-scenarios computed on real-world wind data from Vattenfall AB [20].

As mentioned, interference plays a relevant role in the definition of the problem. Different models have been proposed in the literature to define interference, the most common being kinematic and field models. Those in the former class only consider the velocity deficit of the wake behind a turbine, whereas field models compute the complete flow field through a wind farm. An exhaustive comparison between different models of interference is given in [17]. In the present paper, we consider only the model proposed by Jensen [13] for computing the pairwise interference between a pair of turbines. In addition, we assume the overall interference be the sum of pairwise interferences; though the model is an approximation of the real context, it turns out to be accurate enough for our purposes. Indeed, this model is also used, e.g., in WindPRO, an industry-standard software for wind resource assessment and placement of wind turbines within wind farms [4]. Finally, a main advantage of this model is the possibility to implicitly deal with a large number wind scenarios, which is a must in practical cases. On the contrary, most of the alternative models for interference turn out to be impractical in these settings, as they require the definition of a large number of additional variables and constraints.

Our aim is to heuristically solve wind farm instances of large size, as arise in practical applications. To give the planners a reactive tool for their what-if analyses, almost-optimal solutions should be computed in a matter of minutes on a standard PC—one hour being our time limit even for the largest cases. With this ambitious goal in mind, we investigated a novel approach that combines fast ad-hoc heuristics with a proximity-search [6] refinement procedure based on a compact Mixed-Integer Linear Programming (MIP) model. Computational results on a large benchmark of realistic instances are presented.

The paper is organized as follows. In Sect. 2 we introduce the MIP model used in our computation, which is designed as a compromise between Linear Programming (LP) relaxation tightness and compactness. A very fast ad-hoc heuristic is presented in Sect. 3, while the proximity search framework is outlined in Sect. 4. The overall scheme that combines our ad-hoc and MIP-based

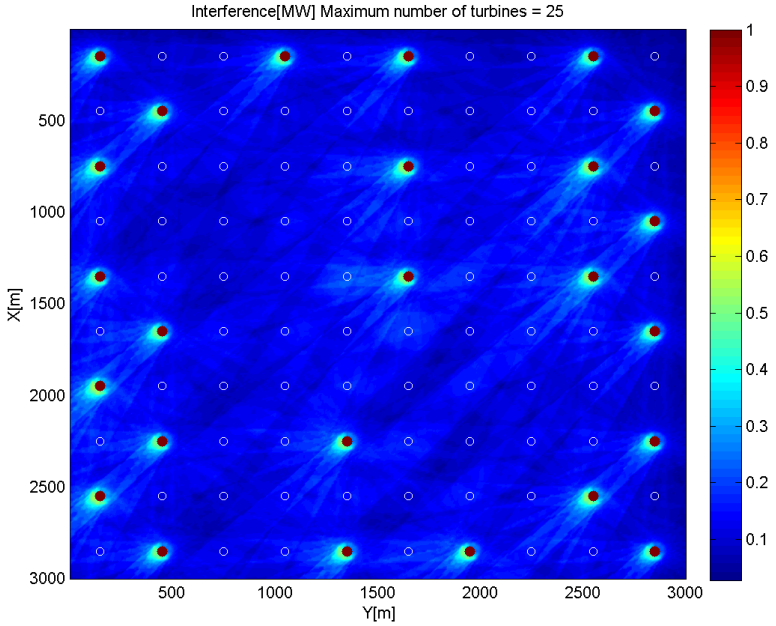

Fig. 1 Turbine packing in an offshore setting with cumulative interference (100 potential turbine locations on a regular 10×10 grid).

heuristics is described in Sect. 5, and computationally evaluated in Sect. 6. Finally, conclusions and directions of future research are addressed in Sect. 7.

The present paper is based on the first author's master thesis [5].

2 Which MIP model?

We consider the problem in which the given offshore area has been sampled, and a number of possible positions for a turbine (called "sites" in what follows) has been identified. Alternative models in which a continuous layout is considered have been proposed in the literature (see, e.g., [14]). However these models are highly nonconvex and turn out to be extremely challenging from a computational viewpoint. Thus, we considered a basic MIP model from the literature, which focuses on turbine distance constraints and on the wake effect (see, e.g., [3]), and addresses the following constraints:

a) a minimum and maximum number of turbines that can be built is given;
b) there should be a minimal separation distance of between two turbines to ensure that the blades do not physically clash (turbine distance constraints);
c) if two turbines are installed, their interference will cause a loss in the power production that depends on their relative position and on wind conditions.

Let V denote the set of possible positions for a turbine, called "sites" in what follows, and let

- I_{ij} be the interference (loss of power) experienced by site j when a turbine is installed at site i, with $I_{jj} = 0$ for all $j \in V$;
- P_i be the power that a turbine would produce if built (alone) at site i;
- N_{MIN} and N_{MAX} be the minimum and maximum number of turbines that can be built, respectively;
- D_{MIN} be the minimum distance between two turbines;
- $dist(i, j)$ be the symmetric distance between sites i and j.

In addition, let $G_I = (V, E_I)$ denote the incompatibility graph with

$$E_I = \{[i, j] : i, j \in V,\, dist(i, j) < D_{MIN},\, j > i\}$$

and let $n := |V|$ denote the total number of sites.

Note that the interference matrix I is not symmetric, as the loss of power due to interference experienced by i when a turbine is installed in site j depends on the relative position of i with respect to j but also on the position of i with respect to the wind direction. In the model, two sets of binary variables are defined: for each $i, j \in V$

$$x_i = \begin{cases} 1 \text{ if a turbine is built at site } i \in V; \\ 0 \text{ otherwise} \end{cases} \qquad (i \in V)$$

$$z_{ij} = \begin{cases} 1 \text{ if two turbines are built at both sites } i \in V \text{ and } j \in V; \\ 0 \text{ otherwise} \end{cases} \qquad (i, j \in V, i < j)$$

The model then reads

$$\max \sum_{i \in V} P_i x_i - \sum_{i \in V} \sum_{j \in V, i<j} (I_{ij} + I_{ji}) z_{ij} \tag{1}$$

$$\text{s.t.} \quad N_{MIN} \le \sum_{i \in V} x_i \le N_{MAX} \tag{2}$$

$$x_i + x_j \le 1 \qquad \forall [i, j] \in E_I \tag{3}$$

$$x_i + x_j - 1 \le z_{ij} \qquad \forall i, j \in V, i < j \tag{4}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{5}$$

$$z_{ij} \in \{0, 1\} \qquad \forall i, j \in V, i < j \tag{6}$$

Objective function (1) maximizes the total power production by taking interference losses I_{ij} into account. Constraints (4) force $z_{ij} = 1$ whenever $x_i = x_j = 1$; because of the objective function, this is in fact equivalent to setting $z_{ij} = x_i x_j$. Constraints (3) model pairwise site incompatibility, and can be strengthened to their clique counterpart

$$\textstyle\sum_{h \in Q} x_h \le 1 \qquad \forall Q \in \mathcal{Q} \tag{7}$$

where $\mathcal{Q}$ is a family of maximal cliques of G_I, such that every edge in E_I is contained in at least one member of $\mathcal{Q}$. Constraints (6) can be relaxed to $z_{ij} \geq 0$, as integrality of the x variables implies the same property for the z.

The definition of the turbine power vector (P_i) and of interference matrix (I_{ij}) depends on the wind scenario considered, which greatly varies in time. Using statistical data, one can in fact collect a large number K of wind scenarios k, each associated with a pair (P^k, I^k) and with a probability π_k. Using that data, one can write a straightforward Stochastic Programming variant of the previous model where only the objective function needs to be modified into

$$\sum_{k=1}^{K} \pi_k \left(\sum_{i \in V} P_i^k x_i - \sum_{i \in V} \sum_{j \in V, i<j} (I_{ij}^k + I_{ji}^k) z_{ij} \right) \tag{8}$$

while all constraints stay unchanged as they only involve "first-stage" variables x and z. It is therefore sufficient to define

$$P_i := \sum_{k=1}^{K} \pi_k P_i^k \qquad \forall i \in V \tag{9}$$

$$I_{ij} := \sum_{k=1}^{K} \pi_k I_{ij}^k \qquad \forall i, j \in V \tag{10}$$

to obtain the same model (1)–(6) as before.

As already mentioned, assuming cumulative interference provides an approximated model, whose accuracy is however quite accurate (see, again, [13]). Though more complex models of interference are available in the literature (see, e.g., [1] and [17]), we decided to stick to the model above for two main reasons: (i) the model is quite standard and well understood by practitioners [3], (ii) as we have just seen, a suitable definition of the input data allows one to easily address the realistic situation in which many wind scenarios are considered; this is not the case for more sophisticated models, which typically lead to really huge stochastic programming variants.

While (1)–(6) turns out to be a reasonable model when just a few sites have to be considered (say $n \approx 100$), it becomes hopeless when $n \geq 1000$ because of the huge number of variables and constraints involved, which grows quadratically with n. Therefore, when facing instances with several thousand of sites an alternative (possibly weaker) model is required, where interference can be handled by a number of variables and constraints that grows just linearly with n. The model below is a compact reformulation of model (1)–(6) that follows a recipe of Glover [10] that is widely used, e.g., in the Quadratic Assignment Problem [21, 7]. The original objective function (to be maximized), written as

$$\sum_{i \in V} P_i x_i - \sum_{i \in V} \left(\sum_{j \in V} I_{ij} x_j \right) x_i \tag{11}$$

is restated as

$$\sum_{i \in V} (P_i x_i - w_i) \tag{12}$$

where

$$w_i := \Big(\sum_{j \in V} I_{ij} x_j \Big) x_i = \begin{cases} \sum_{j \in V} I_{ij} x_j & \text{if } x_i = 1; \\ 0 & \text{if } x_i = 0. \end{cases}$$

denotes the total interference caused by site i. Our compact model then reads

$$\max z = \sum_{i \in V} (P_i x_i - w_i) \tag{13}$$

$$\text{s.t.} \quad N_{MIN} \leq \sum_{i \in V} x_i \leq N_{MAX} \tag{14}$$

$$x_i + x_j \leq 1 \qquad \forall [i,j] \in E_I \tag{15}$$

$$\sum_{j \in V} I_{ij} x_j \leq w_i + M_i (1 - x_i) \qquad \forall i \in V \tag{16}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{17}$$

$$w_i \geq 0 \qquad \forall i \in V \tag{18}$$

where the big-M term $M_i = \sum_{\substack{j \in V \\ [i,j] \notin E_I}} I_{ij}$ is used to deactivate constraint (16) in case $x_i = 0$.

Our preliminary tests suggested not to explicitly strengthen constraints (15) to their clique form (7), as a family of cliques is automatically generated during preprocessing by the MIP solver in a very efficient way.

3 Which ad-hoc heuristic?

A simple 1- and 2-opt heuristic with local-minimum escape through fictitious bounds on the turbine number was implemented. Other simple heuristics (including tabu search) have been tried but seem to have a worse performance, at least in our implementation.

The core of our heuristic is a parametrized 1-opt search. At each step, we have an incumbent solution, say $\tilde{x}$, that describes the best-known turbine allocation ($\tilde{x}_i = 1$ if a turbine is built at site i, 0 otherwise), and a current solution x. Let

$$z = \sum_{i \in V} P_i x_i - \sum_{i \in V} \sum_{j \in V} I_{ij}\, x_i\, x_j$$

be the profit of the current solution,

$$\gamma = \sum_{i \in V} x_i$$

be its cardinality, and define for each $j \in V$ the extra-profit δ_j incurred when flipping x_j, namely:

$$\delta_j = \begin{cases} P_j - \sum_{i \in V : x_i = 1} (I_{ij} + I_{ji}) & \text{if } x_j = 0; \\ -P_j + \sum_{i \in V : x_i = 1} (I_{ij} + I_{ji}) & \text{if } x_j = 1 \end{cases}$$

where we assume $I_{ij} = BIG$ for all incompatible pairs $[i, j] \in E_I$, and BIG is a large penalty value (e.g., $BIG > \sum_{i \in V} P_i$), while $I_{ii} = 0$ as usual.

We start with $x = 0$, $z = 0$, $\gamma = 0$ and initialize $\delta_j = P_j$ for all $j \in V$. We also define a local copy of N_{MIN} and N_{MAX}, say n_1 and n_2. Then, we iteratively improve x by a sequence of 1-opt moves, according to the following scheme. At each iteration, we look in $O(n)$ time for the site j with maximum $\delta_j + FLIP(j)$, where function $FLIP(j)$ takes cardinality constraints into account, namely

$$FLIP(j) = \begin{cases} -HUGE & \text{if } x_j = 0 \text{ and } \gamma \geq n_2 \\ -HUGE & \text{if } x_j = 1 \text{ and } \gamma \leq n_1 \\ +HUGE & \text{if } x_j = 0 \text{ and } \gamma < n_1 \\ +HUGE & \text{if } x_j = 1 \text{ and } \gamma > n_2 \\ 0 & \text{otherwise} \end{cases}$$

with $HUGE >> BIG$ (recall that the function $\delta_j + FLIP_j$ has to be maximized). In our implementation we used $BIG = 10,000$ and $HUGE = 1,000,000$. Once the best j has been found, say $j = j^*$, if $\delta_{j^*} + FLIP(j^*) > 0$ we just flip x_{j^*}, update x, z, and γ in $O(1)$ time, update all δ_j's in $O(n)$ time (as explained below), and repeat. In this way a sequence of improving solutions x (and hence $\tilde{x}$) is obtained, a local optimal solution x that cannot be improved by just one flip is found.

As to time complexity, the most time consuming step is the update of each δ_j as a result of the flip of a single x_{j^*}. However, each update requires just $O(1)$ time through the following parametrized formula, to be applied *before* the flip of x_{j^*}:

$$\delta_j = \begin{cases} -\delta_j & \text{if } j = j^* \\ \delta_j - (I_{jj^*} + I_{j^*j}) & \text{if } j \neq j^* \text{ and } x_j = x_{j^*} \\ \delta_j + (I_{jj^*} + I_{j^*j}) & \text{if } j \neq j^* \text{ and } x_j \neq x_{j^*} \end{cases}$$

Validity of the above formula is obvious for $j = j^*$, whereas for the other cases it follows directly from the definition of δ_j by considering the four combinations $(x_j, x_{j^*}) \in \{(0,0), (0,1), (1,0), (1,1)\}$. It then follows that each 1-opt iteration requires $O(n)$ time, as claimed, whereas a non-parametric implementation would require $O(n^2)$ time.

To escape local minima, a number of metaheuristic approaches can be used, e.g., Tabu Search [11] or Variable Neighborhhod Search [15]. In our implementation, we used an alternative scheme that produced good results for our instances. The idea is to modify the local limits n_1 and n_2 to force the

current x to move to fulfill them, thus visiting different parts of the solution space. More specifically, as soon as we get to a local minimum (i.e., $\delta_{j^*} + FLIP(j^*) \leq 0$), we generate a uniformly pseudo-random value $\rho \in [0, 1]$ and update the local limits as follows:

$$n_1 := n_2 := \begin{cases} \gamma(1 + \rho/2) + 10 & \text{if } \gamma \leq \sum_{i \in V} \tilde{x}_i \\ \gamma(1 - \rho/2) - 10 & \text{otherwise} \end{cases}$$

In this way we obtain an oscillatory behavior where the cardinality of the current x (namely, γ) goes up and down, following a zig-zag trajectory. Each time a new solution x is constructed, the incumbent $\tilde{x}$ is possibly updated by considering the true limits N_{MIN} and N_{MAX} (instead of their local counterparts n_1 and n_2).

In our algorithm, we also apply a sequence of improving 2-opt exchanges to the current solution x, until no improving 2-opt exchange exists. This step is useful as it allows, e.g., to move a single turbine to a nearby (better) site. As each 2-opt exchange requires $O(n^2)$ time, however, this phase is applied in a conservative way, also because it interferes with the zig-zag mechanism and tends to produce worse solutions in the long run. In our implementation, improving 2-opt exchanges on x are only applied immediately before a change of the local limits n_1 and n_2, and on the final incumbent $\tilde{x}$, just before it is returned.

The above heuristic is applied in two different modes. In the "initial solution" mode, we start with $\tilde{x} := x := 0$ and repeat the procedure until we count a very large number (10,000) of consecutive 1-opt calls with no improvement of $\tilde{x}$. In the faster "clean-up" mode, instead, we already have an incumbent $\tilde{x}$ to refine, so we initialize $x := \tilde{x}$ and repeat the procedure until we count 100 consecutive 1-opt calls with no improvement of $\tilde{x}$. As already mentioned, 2-opt exchanges are applied in all cases before the final $\tilde{x}$ is returned.

4 Which MIP heuristic?

We now address how to improve a given feasible solution $(\tilde{x}, \tilde{w})$ by exploiting MIP model (13)–(18). One standard option would be to just use $(\tilde{x}, \tilde{w})$ to initialize the incumbent solution of the MIP solver, and to run it in its default mode. However, it is common experience that this strategy is unlikely to produce improved solutions within acceptable computing times, the main so if the underlying MIP model is very large and the formulation is weak—as it happens in out context. So, we preferred to address a different use of the MIP solver, to be applied to "search a neighborhood" of $(\tilde{x}, \tilde{w})$. In particular, our algorithm belongs to the *Large Neighborhood Search* scheme (see, e.g., [18], [9] and [16]), as we consider an exponentially large neighborhood and explore it using the *proximity search* strategy recently proposed in [6], that seems particularly suited for models involving big-M constraints.

Proximity search works in stages, each aimed at producing an improved feasible solution, and is illustrated in Figure 2. At each stage, an explicit cutoff

constraint

$$\sum_{i \in V}(P_i x_i - w_i) \geq \sum_{i \in V}(P_i \tilde{x}_i - \tilde{w}_i) + \theta \tag{19}$$

is added to the original MIP, where $\theta > 0$ is a given tolerance that specifies the minimum improvement required. The objective function of the problem can then be replaced by a new "proximity function" (to be minimized):

$$\Delta(x, \tilde{x}) = \sum_{j \in V:\, \tilde{x}_j = 0} x_j + \sum_{j \in V:\, \tilde{x}_j = 1} (1 - x_j) \tag{20}$$

that measures the Hamming distance between a generic binary x and the given $\tilde{x}$; note that continuous variables w_i's play no role in this definition. One then applies the MIP solver, as a black box, to the modified problem in the hope of finding an improved solution at a small Hamming distance from $\tilde{x}$. The computational experience reported in [6] confirms that this approach is quite successful (at least, on some classes of problems), due to the action of the proximity objective function that is beneficial both in speeding up the solution of the LP relaxations, and in driving the heuristics embedded in the MIP solvers—thus resulting into an improved "relaxation grip" [6].

Proximity search:
let $(\tilde{x}, \tilde{w})$ be the initial feasible solution to improve;
repeat
  explicitly add cutoff constraint (19) to the MIP model;
  install the new "proximity" objective function (20) to be minimized;
  run the MIP solver on the new model until a termination condition
  is reached, and let (x^*, w^*) be the best feasible solution found;
  refine w^* by solving the original model (13)–(18) after fixing $x = x^*$;
  recenter $\Delta(x, \cdot)$ by setting $\tilde{x} := x^*$, and/or update θ
until *an overall termination condition is reached*;
return $(\tilde{x}, \tilde{w})$

Fig. 2 The basic proximity search scheme

In our implementation, we used an improved version of the above scheme, called "proximity search with an incumbent" in [6]. The idea is that one would like to provide the MIP-solver an incumbent by using the current solution $(\tilde{x}, \tilde{w})$, which is however infeasible because of the cutoff constraint. So, one can introduce a continuous variable $\xi \geq 0$ and weaken (19) to its "soft" version:

$$\sum_{i \in V}(P_i x_i - w_i) \geq \sum_{i \in V}(P_i \tilde{x}_i - \tilde{w}_i) + \theta(1 - \xi) \tag{21}$$

while minimizing $\Delta(x, \tilde{x}) + U\xi$ instead of just $\Delta(x, \tilde{x})$, where $U >> 0$ is a very large value with respect to Δ; see again [6] for details.

5 The overall approach

As already mentioned, our approach can be cast into the *Large Neighborhood Search* paradigm, and in particular in the *MIP-and-refine* framework recently investigated in [8], and works as shown in Figure 3.

Step 0. read input data and compute the overall interference matrix (I_{ij});
Step 1. apply ad-hoc heuristics (iterated 1-opt) to get a first incumbent $\tilde{x}$;
Step 2. apply quick ad-hoc refinement heuristics (few iterations of iterated 1- and 2-opt) to possibly improve $\tilde{x}$;
Step 3. if $n > 2000$, randomly remove points $i \in V$ with $\tilde{x}_i = 0$ so as to reduce the number of candidate sites to 2000;
Step 4. build a MIP model for the resulting subproblem and apply proximity search to refine $\tilde{x}$ until the very first improved solution is found (or time limit is reached);
Step 5. if time limit permits, repeat from Step 2.

Fig. 3 Our overall heuristic framework

At Step 1. (respectively, Step 2.) the ad-hoc heuristic of Section 3 is applied in its initial-solution (resp., clean-up) mode. Two different MIP models are used to feed the proximity-search heuristic at Step 4. During the first part of the computation, we use a simplified MIP model obtained from (13)–(18) by removing all interference constraints (16), thus obtaining a much easier relaxation. A short time limit (60 sec.s) is imposed for each call of proximity search when this simplified model is solved. In this way we aggressively drive the solution $\tilde{x}$ to increase the number of built turbines, without being bothered by interference considerations and only taking pairwise incompatibility (15) into account. This approach quickly finds better and better solutions (even in terms of the true profit), until either (i) no additional turbine can be built, or (ii) the addition of new turbines does in fact reduce the true profit associated to the new solution. In this situation we switch to the complete model (13)–(18) with all interference constraints, which is used in all next executions of Step 4. Note that the simplified model is only used at Step 4, while all other steps of the procedure always use the true objective function that takes interference into full account.

6 Computational results

The following alternative solution approaches were implemented in C language, some of which using the commercial MIP solver IBM ILOG Cplex 12.5.1 [12]; because of the big-M's involved in the models, all Cplex's codes use zero as integrality tolerance (`CPX_PARAM_EPINT` = 0.0).

a) `proxy`: our MIP-and-refine heuristic, as outlined in the previous section, using Cplex with the following aggressive parameter tuning: all cuts deactivated, `CPX_PARAM_RINSHEUR` = 1, `CPX_PARAM_POLISHAFTERTIME` = 0.0, `CPX_PARAM_INTSOLLIM` = 2;

b) `cpx_def`: the application of IBM ILOG Cplex 12.5.1 in its default setting, starting from the same heuristic solution $\tilde{x}$ found by `proxy` after the first execution of Step 2 of Figure 3;
c) `cpx_heu`: same as `cpx_def`, with the following internal tuning intended to improve Cplex's heuristic performance: all cuts deactivated, `CPX_PARAM_RINSHEUR` = 100, `CPX_PARAM_POLISHAFTERTIME` = 20% of the total time limit;
d) `loc_sea`: a simple local-search procedure not based on any MIP solver, that just loops on Steps 2 of Figure 3 and randomly removes installed turbines from the current best solution after 10,000 iterations without improvement of the incumbent.

For each algorithm, we considered the best solution found within a given time limit. In our view, `loc_sea` is representative of a clever but not oversophisticated metaheuristic, as typically implemented in practice, while `cpx_def` and `cpx_heu` represent a standard way of exploiting a MIP model once a good feasible solution is known.

Our testbed refers to an offshore 3,000 × 3,000 (m) square with $D_{MIN} =$ 400 (m) minimum turbine separation, with no limit on the number of turbines to be built (i.e., $N_{MIN} = 0$ and $N_{MAX} = +\infty$). Turbines are all of Siemens SWT-2.3-93 type (diameter 93m), which produces a power of 0.0 MW for wind speed up to 3 m/s, of 2.3 MW for wind speed greater than or equal to 16 m/s, and intermediate values for winds in range 3-16 m/s according to a nonlinear function [19]. Pairwise interference (in MW) was computed using Jensen's model [13], by averaging 250,000+ real-world wind samples. Those samples were grouped into about 500 macro-scenarios to reduce the computational time spent defining the interference matrix. A pairwise average interference of 0.01 MW or less is treated as zero. The reader is referred to [5] for details.

We generated five classes of medium-to-large problems with $n = 1000$, 5000, 10000, 15000, and 20000. For each class, 10 instances have been considered by generating n uniformly random points in the 3,000 × 3,000 square. (Although in the offshore case turbine positions are typically sampled on a regular grid, we decided to randomly generate them to be able to compute meaningful statistics for each value of n.)

In what follows, reported computing times are in CPU seconds of an Intel Xeon E3-1220 V2 quad-core PC with 16GB of RAM, and do not take Step 0 of Figure 3 into account as the interference matrix is assumed to be precomputed and reused at each what-if analysis run.

Computational results on our instances are given in Table 1, where each entry refers to the performance of a given algorithm at a given time limit. In particular, the left part of the table reports, for each algorithm and time limit, the *number of wins*, i.e, the number of instances for which a certain algorithm produced the best solution at the given time limit (ties allowed).

According to the table, `proxy` outperforms all competitors by a large amount for medium to large instances. As expected, `cpx_heu` performs better for instances with $n = 1{,}000$ as it is allowed to explore a large number of

enumeration nodes for the original model and objective function. Note that `loc_sea` has a good performance for short time limits and/or for large instances, thus confirming its effectiveness, whereas `cpx_heu` is significantly better than `loc_sea` only for small instances and large time limits.

A different performance measure is given in the right-hand side part of Table 1, where each entry gives the *average optimality ratio*, i.e., the average value of the ratio between the solution produced by an algorithm (on a given instance at a given time limit) and the best solution known for that instance—the closer to one the better. It should be observed that an improvement of just 1% has a very significant economical impact due to the very large profits involved in the wind farm context. The results show that `proxy` is always able to produce solutions that are quite close to the best one. As before, `loc_sea` is competitive for large instances when a very small computing time is allowed, whereas `cpx_def` and `cpx_heu` exhibit a good performance only for small instances, and are dominated even by `loc_sea` for larger ones.

Figure 4 plots the incumbent value (i.e., the profit of the current best solution) over CPU time for the four heuristics under comparison, and refer to 4 sample instances in our testbed. The two subfigures on the top refer to two small instances with $n = 1{,}000$, where `proxy`, `cpx_heu` and `cpx_def` have a comparable performance and clearly outperform `loc_sea`. For $n = 5{,}000$ (bottom-left subfigure) and $n = 10{,}000$ (bottom-right subfigure), however, both `cpx_def` and `cpx_heu` (and also `loc_sea`) have hard time in improving their initial solution, and are outperformed by `proxy` by a large amount.

7 Conclusions

We have considered an important practical problem in wind farm optimization, namely, the optimal allocation of turbines subject to interference conditions. Our goal was the design of a fast heuristic capable of handling instances with 10,000+ potential sites in a matter a minutes. To this end, we have exploited two basic tools: a fast ad-hoc heuristic, and a MIP model designed for the very large instances of interest. A synergic use of these two tools has been proposed, following a clever MIP-and-refine recipe where two different variants of the underlying MIP model have been solved through a proximity search heuristic. Computational results on a testbed of medium-to-large scale instances have shown that the approach outperforms a standard use of the two basic tools.

A lesson learned is that the choice of the MIP model to be used is a critical step in the design of the overall heuristic framework, because an effective compromise between tightness and compactness is required. In particular, models that are considered weak when solving small instances to proven optimality can become effective when used in a refining mode for large instances. In addition, simplified MIP models that relax some details of the problem (the effect of interference, in our case) can be very useful at the early stage of the heuristic.

Table 1 Number of times each algorithm finds the best solution within the time limit (wins), and optimality ratio with respect to the best known solution—the larger the better.

		number of wins				optimality ratio			
n	Time limit (s)	proxy	cpx_def	cpx_heu	loc_sea	proxy	cpx_def	cpx_heu	loc_sea
1,000	60	6	1	3	0	0.994	0.983	0.987	0.916
	300	4	2	4	0	0.997	0.991	0.998	0.922
	600	7	3	7	0	0.997	0.992	0.997	0.932
	900	5	2	3	0	0.998	0.993	0.996	0.935
	1,200	5	1	5	0	0.998	0.992	0.997	0.939
	1,800	5	1	4	0	0.998	0.992	0.996	0.942
	3,600	4	2	5	0	0.998	0.995	0.997	0.943
5,000	60	9	6	6	5	0.909	0.901	0.901	0.904
	300	10	0	0	0	0.992	0.908	0.908	0.925
	600	10	0	10	0	0.994	0.908	0.994	0.935
	900	10	0	0	0	0.994	0.908	0.908	0.936
	1,200	10	0	0	0	0.994	0.908	0.925	0.939
	1,800	9	0	1	0	0.996	0.908	0.971	0.946
	3,600	5	0	5	0	0.996	0.932	0.994	0.948
10,000	60	9	9	8	10	0.914	0.913	0.914	0.914
	300	10	2	2	2	0.967	0.927	0.927	0.936
	600	10	0	10	0	0.998	0.928	0.998	0.944
	900	10	0	0	0	1.000	0.928	0.928	0.948
	1,200	10	0	0	0	1.000	0.928	0.928	0.951
	1,800	10	0	0	0	1.000	0.928	0.928	0.957
	3,600	9	0	0	1	1.000	0.928	0.928	0.964
15,000	60	9	10	9	9	0.909	0.912	0.911	0.909
	300	10	8	7	8	0.943	0.937	0.935	0.937
	600	10	0	10	0	0.992	0.939	0.992	0.942
	900	10	0	0	0	1.000	0.939	0.939	0.956
	1,200	9	0	0	1	1.000	0.939	0.939	0.959
	1,800	9	0	0	1	1.000	0.939	0.939	0.965
	3,600	9	0	0	1	1.000	0.939	0.939	0.972
20,000	60	9	9	9	10	0.901	0.902	0.901	0.902
	300	10	8	10	10	0.933	0.933	0.933	0.933
	600	9	0	9	1	0.956	0.935	0.956	0.941
	900	10	0	0	0	0.978	0.935	0.935	0.945
	1,200	10	0	0	0	0.991	0.935	0.935	0.950
	1,800	10	0	0	0	0.999	0.935	0.935	0.963
	3,600	9	0	0	0	1.000	0.935	0.935	0.971
ALL	60	42	35	35	34	0.925	0.922	0.922	0.909
	300	44	20	23	20	0.966	0.939	0.940	0.930
	600	46	3	46	1	0.987	0.941	0.987	0.938
	900	45	2	3	0	0.994	0.941	0.941	0.944
	1,200	44	1	5	1	0.997	0.940	0.945	0.947
	1,800	43	1	5	1	0.999	0.940	0.954	0.955
	3,600	36	2	10	2	0.999	0.946	0.959	0.959

Future research should evaluate different ways to sparsify the problem by removing candidate sites (Step 3 of Figure 3). In our runs we used a simple random criterion, but more clever options that favor the removal of points far from all installed turbines are also possible. By putting this mechanism to its extreme extent, it is in fact conceivable to address a "continuous" version of the problem where a turbine can be installed at any points in a certain geographical area, and the heuristic dynamically discretizes it by generating and removing sites $i \in V$ on the fly.

Acknowledgements The research of the second author was supported by Miur (project PRIN 2012) and by the University of Padova (Progetto di Ateneo "Exploiting randomness in

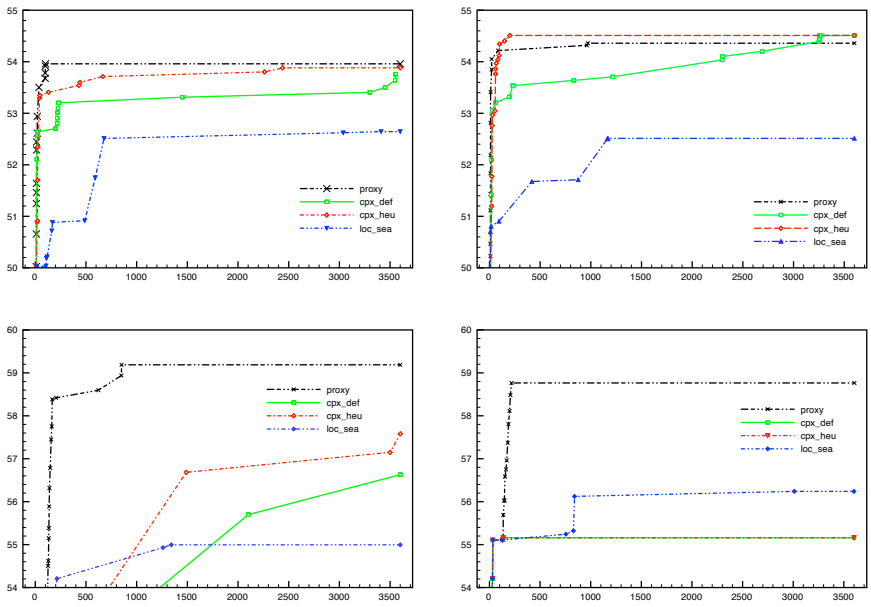

Fig. 4 Solution profit over time for 4 sample instances with $n = 1{,}000$ (top left and top right), $n = 5000$ (bottom left), and $n = 10{,}000$ (bottom right); the higher the profit the better.

Mixed Integer Linear Programming"). We thank Jakob Stoustrup and John Leth (Automation and Control, Department of Electronic Systems, Aalborg University) who supervised the thesis work of first author in Aalborg, and Benjamin Martinez (R&D Wind Engineer, Vattenfall AB) who provided the real-world wind scenarios used in our tests. Thanks are also due to three anonymous referees for their helpful comments.

References

1. Archer, R., Nates, G., Donovan, S., Waterer, H.: Wind turbine interference in a wind farm layout optimization mixed integer linear programming model. Wind Engineering **35**, 165–178 (2011)
2. Barthelmie, R., Hansen, K., Frandsen, S.T., Rathmann, O., Schepers, J., Schlez, W., Phillips, J., Rados, K., Zervos, A., Politis, E., Chaviaropoulos, P.K.: Modelling and measuring flow and wind turbine wakes in large wind farms offshore. Wind Energy **12**, 431–444 (2009)
3. Donovan, S.: Wind farm optimization. In: Proceedings of the 40th Annual ORSNZ Conference, pp. 196–205 (2005)
4. EMD: WindPRO. `http://www.emd.dk/windpro/frontpage`
5. Fischetti, M.: Mixed-integer models and algorithms for wind farm layout optimization. Master's thesis, University of Padova (2014). `http://tesi.cab.unipd.it/45458/1/tesi_Fischetti.pdf`
6. Fischetti, M., Monaci, M.: Proximity search for 0-1 mixed-integer convex programming. Journal of Heuristics **20**(6), 709–731 (2014)
7. Fischetti, M., Monaci, M., Salvagnin, D.: Three ideas for the quadratic assignment problem. Operations Research **60**(4), 954–964 (2012)

8. Fischetti, M., Sartor, G., Zanette, A.: A MIP-and-refine matheuristic for smart grid energy management. International Transactions in Operational Research **22**(1), 49–59 (2015)
9. Focacci, F. Laburthe, F., Lodi, A.: Local Search and Constraint Programming. In: Handbook of Metaheuristics Management Science vol. **57**, F. Glover and G.A. Kochenberger (Eds.), pp. 369–403 (2003)
10. Glover, F.: Improved linear integer programming formulations of nonlinear integer problems. Management Science **22**, 455–460 (1975)
11. Glover, F.: Tabu search: A tutorial. Interfaces **20**(4), 74–94 (1990)
12. IBM ILOG CPLEX: Optimization Studio (2013). `http://www.cplex.com`
13. Jensen, N.: A note on wind generator interaction. Tech. rep., Technical Report Riso-M-2411(EN), Riso National Laboratory, Roskilde, Denmark (1983)
14. Kusiak, A., Song, Z.: Design of wind farm layout for maximum wind energy capture. Renewable Energy **35**, 685–694 (2010)
15. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research **24**(11), 1097–1100 (1997)
16. Pisinger, D., Ropke, S.: Large Neighborhood Search, vol. 146, pp. 399–419 (2010)
17. Renkema, D.: Validation of wind turbine wake models. Master's thesis, Delft University of Technology (2007)
18. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: 4th International Conference CP98, Lecture Notes in Computes Science **1520**. M. Maher and J.F. Puget (eds.), pp. 417–431 (1998)
19. Siemens AG: SWT-2.3-93 Turbine, Technical Specifications. `http://www.energy.siemens.com`
20. Vattenfall AB. Personal communication (2014)
21. Xia, Y., Yuan, Y.: A new linearization method for quadratic assignment problem. Optimization Methods and Software **21**, 803–816 (2006)