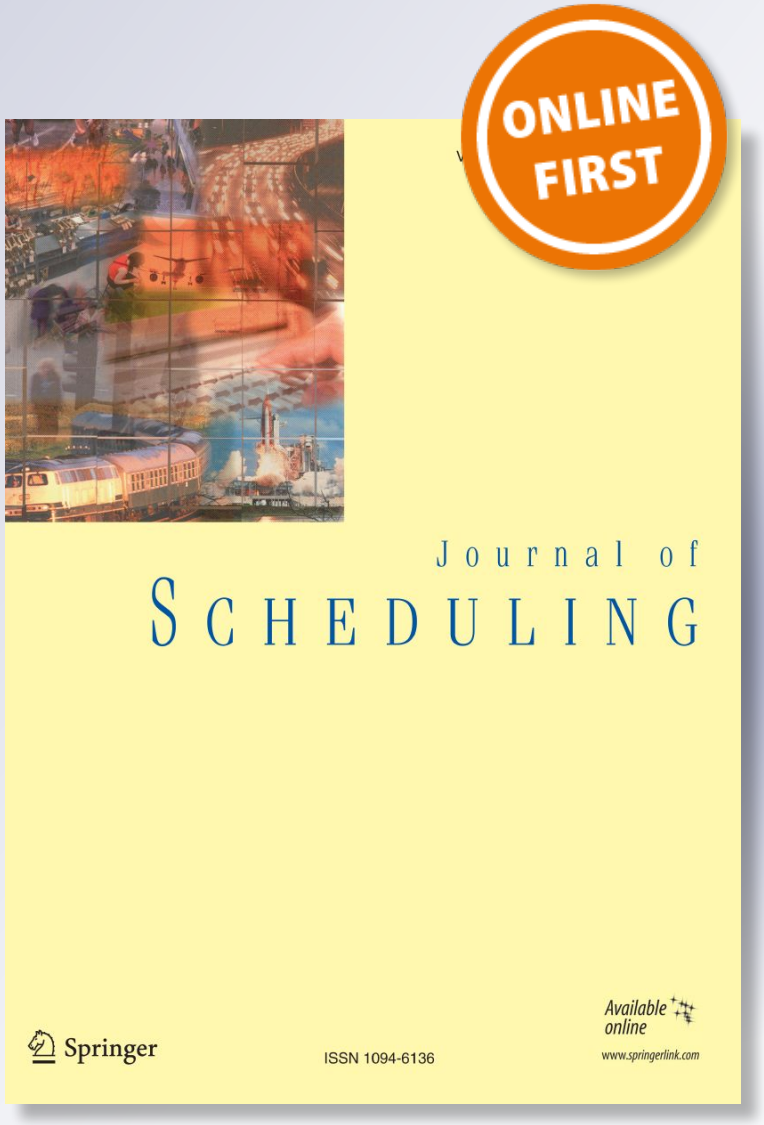
Solving a wind turbine maintenance scheduling problem

Aurélien Froger, Michel Gendreau, Jorge E. Mendoza, Eric Pinson & Louis-Martin Rousseau

Journal of Scheduling

ISSN 1094-6136

J Sched
DOI 10.1007/s10951-017-0513-5



Springer

J Sched
DOI 10.1007/s10951-017-0513-5

Solving a wind turbine maintenance scheduling problem

Aurélien Froger[1] · **Michel Gendreau**[2] · **Jorge E. Mendoza**[3,4] · **Eric Pinson**[1] · **Louis-Martin Rousseau**[2]

Abstract Driven by climate change mitigation efforts, the wind energy industry has significantly increased in recent years. In this context, it is essential to make its exploitation cost-effective. Maintenance of wind turbines therefore plays an essential role in reducing breakdowns and ensuring high productivity levels. In this paper, we discuss a challenging maintenance scheduling problem rising in the onshore wind power industry. While the research in the field primarily focuses on condition-based maintenance strategies, we aim to address the problem on a short-term horizon considering the wind speed forecast and a fine-grained resource management. The objective is to find a maintenance plan that maximizes the revenue from the electricity production of the turbines while taking into account multiple task execution modes and task-technician assignment constraints. To solve this problem, we propose a constraint programming-based large neighborhood search (CPLNS) approach. We also propose two integer linear programming formulations that we solve using a commercial solver. We report results on randomly generated instances built with input from wind forecasting and maintenance scheduling software companies. The CPLNS shows an average gap of 1.2% with respect to the optimal solutions if known, or to the best upper bounds otherwise. These computational results demonstrate the overall efficiency of the proposed metaheuristic.

Keywords Maintenance · Scheduling · Large neighborhood search · Constraint programming

✉ Aurélien Froger
aurelien.froger@uco.fr

Michel Gendreau
michel.gendreau@cirrelt.ca

Jorge E. Mendoza
jorge.mendoza@univ-tours.fr

Eric Pinson
eric.pinson@uco.fr

Louis-Martin Rousseau
louis-martin.rousseau@cirrelt.net

1 Université Bretagne Loire, Université Catholique de l'Ouest, LARIS EA 7315, Angers, France

2 CIRRELT Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, and Département de mathématiques et de génie industriel, Polytechnique Montréal, Montreal, Canada

3 CNRS, LI EA 6300, ROOT ERL CNRS 6305, Université François-Rabelais de Tours, Tours, France

4 Centre de Recherches Mathématiques (UMI 3457 CNRS), Montreal, Canada

1 Introduction

With a 63-gigawatt (GW) increase in the global installed capacity in 2015 (and a total of about 432 GW), wind energy is currently the world's fastest growing source of electricity.[1] Boosted by the ever-increasing environment awareness and the constantly decreasing cost of turbines, wind power is expected to account for up to 20% of the global electricity production by 2050[1] (vs. 2.4% in 2015). The Paris Agreement (resulting from the 2015 United Nations Climate Change Conference—COP21) is in this respect a clear evidence that the renewable energy sector will keep growing in order to reduce greenhouse gas emissions. In this context, efficient wind turbine maintenance planning and scheduling

[1] The Global Wind Energy Council—Global Wind Report Annual Market Update 2015—http://www.gwec.net/wp-content/uploads/vip/GWEC-Global-Wind-2015-Report_April-2016_22_04.pdf, last accessed: 2016-09-15.

Published online: 17 March 2017

become a critical tool to prevent unnecessary downtime and excessive operational costs.

Maintenance planning and scheduling have been widely studied in different industrial contexts (see, for example, Budai et al. 2008 for a survey). In general, however, solutions remain sector-specific. In the particular case of traditional electricity generation plants, the problem is concerned with the definition of time intervals for preventive maintenance of generating units under financial (cost minimization, profits maximization) and/or reliability (leveling, maximization of the net reserves) considerations. By now, the literature reports on a number of solution approaches to tackle these problems. We refer the reader to Froger et al. (2016) for a comprehensive review. Unfortunately, these approaches are inapplicable to the wind power industry. One of the main reasons is that wind farms are usually owned by investment funds, and the operation and the maintenance of the turbines are often outsourced to a third party. As it stands, the stakeholders and the contractors may potentially face conflicting objectives: maximize energy production versus minimize maintenance costs. Therefore, service contracts are set between these two entities. They include incentives and penalties if some target values (on the production and/or the availability factor[2] of wind turbines) are reached or not. Another specificity of the wind power industry is that maintenance decisions are not correlated with the electricity demand, since producers are mostly not required to satisfy production goals fixed in advance. The objective then tends to be the maximization of the efficiency of the wind turbines. Last but not least, the wind power production is inherently volatile, and the meteorological conditions have a great impact on the maintenance plan and can induce last-minute adjustments. In summary, the aim of maintenance companies is to schedule the maintenance in order to meet their contract commitments. Although sometimes it is not their top priority, producing maintenance plans for which the production of the turbines is maximized, while taking into consideration their internal constraints, is a meaningful strategy to avoid interference with the stakeholders and to potentially increase their revenue. If the maintenance is not outsourced, this objective is all the more relevant.

Maintenance optimization for wind turbines has only recently started to received attention in the literature (we refer the reader to Ding et al. 2013 for a survey). This stream of research primarily focuses on the definition of maintenance policies according to failure models or/and condition monitoring. Although existing studies precisely define time intervals during which the maintenance has to be performed in order to reduce the loss of energy production, they do not consider a fine-grained resource management. Therefore, the obtained results are used more as guidelines to define maintenance time windows, than as an actual maintenance plan. In this regard, they can be used to set the service contracts (e.g., preventive maintenance has to be performed every 6 months on each turbine).

Fine-grained resource management implies, among others, considering a multi-skilled workforce, coping with individual or global resource unavailability time periods (e.g., vacations) and taking into account resource location-based constraints. Dealing with these issues requires considering a short-term planning horizon. In this context, existing studies allow planners to define the tasks to be performed during the planning horizon and to set the maintenance time window constraints. Nonetheless, the maintenance scheduling problem still contains a degree of operational flexibility. Considering fine-grained resource management then aims to build detailed maintenance plans that can be used on a daily or weekly basis. These latter provide more accurate estimates of turbine downtimes and loss of production, two metrics that can otherwise be underestimated, which may lead to significant prediction errors. Indeed, producing a maintenance plan in which no operations generate a loss of production (e.g., is scheduled during time periods where the wind speed is below $4\,\mathrm{m\,s^{-1}}$, which is too low to produce electricity) can almost never be achieved in practice, since human resources are a major bottleneck.

To our knowledge, Kovács et al. (2011) is the only study considering fine-grained resource management while scheduling maintenance operations in onshore wind farms on a 1-day time horizon. These authors aimed to minimize lost production due to maintenance and failures. They introduced incompatibilities between pairs of tasks and managed the assignment of teams of skilled workers to tasks. They modeled the problem as an integer linear program and solved it using a commercial solver. They performed experiments on instances with up to 50 tasks.

In this paper, we introduce a maintenance scheduling problem with resource management rising in the onshore wind power industry. Our problem differs from that introduced by Kovács et al. (2011) in several ways. First, we manage resources (i.e., technicians) individually rather than by teams. Second, we consider multiple task execution modes that impact the task duration as well as the resource requirements (Reyck et al. 1998). Third, we present an alternative way to consider technician transfer times by introducing location-based incompatibility constraints between tasks. The objective of this new problem is to maximize the revenue generated by the total power production of the wind turbines. The work targets a short-term horizon as wind predictions can only be reliably established few days in advance.

The contributions of this paper are twofold. First, we introduce a new maintenance scheduling problem that is especially relevant to the onshore wind power industry. We

[2] The ratio of the duration that a generating unit is available to provide energy to the grid, for the time considered, to the duration of the same time period.

formally define our problem using two different programming paradigms, namely integer linear programming (ILP) and constraint programming (CP). Second, we introduce a constraint programming-based large neighborhood search to efficiently tackle the problem. The proposed approach uses destruction operators either specifically conceived for the problem or adapted from the literature. The repair operator consists in solving a CP model with some fixed variables using branching strategies specially tailored for the problem. We report computational results on randomly generated instances with up to 80 tasks, 3 different skills and 40-period time horizons.

The remainder of this paper is organized as follows. In Sect. 2, we describe the problem. In Sects. 3 and 4, we introduce two integer linear programming formulations and a constraint programming formulation for the problem. In Sect. 5, we present our constraint programming-based large neighborhood search approach. In Sect. 6, we report and discuss computational experiments. In Sect. 7, we describe the handling of corrective tasks. Finally in Sect. 8, we present our conclusions and outline research perspectives.

2 Problem statement

The maintenance scheduling problem we consider consists in scheduling a set $\mathcal{I}$ of tasks during a finite time horizon $\mathcal{T}$ in order to maximize the revenue from the electricity production of a set $\mathcal{W}$ of wind turbines. These wind turbines are geographically distributed over a set $\mathcal{L}$ of locations. We denote $l_w \in \mathcal{L}$ the location of wind turbine $w \in \mathcal{W}$. Each task $i \in \mathcal{I}$ is also associated with a specific location, denoted as l_i.

The time horizon is partitioned in periods of identical length and spans over several days from a set $\mathcal{D}$. We denote $\mathcal{T}_d$ the set of time periods covered by day $d \in \mathcal{D}$. Moreover, since the execution of a task can impact the production during non-working hours, we introduce a special time period (hereafter referred to as a *rest time period*) between two consecutive days to represent, for example, a night or a weekend. Maintenance tasks are non-preemptive, but, obviously, they are interrupted during rest time periods when overlapping consecutive days (e.g., a technician can start a task at the end of 1 day and complete it at the beginning of the next day).

Although we do not include rest time periods in $\mathcal{T}$, we count in the objective function the loss of revenue generated by tasks overlapping these specific time periods. More specifically, tasks may have different impact on the availability of the turbines. Some tasks shut down one (or more) turbine(s) since the task starts until the task ends. For instance, during the maintenance of a wind farm's substation[3] no turbines

[3] A wind farm substation collects the electricity produced by all the turbines of the farm and distributes it through the grid.

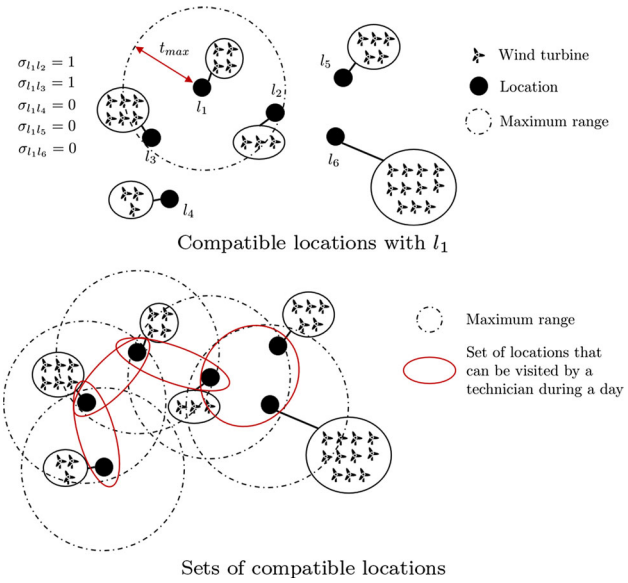


Fig. 1 Illustration of the daily location-based incompatibilities

in the farm can produce electricity. It should be noted, however, that tasks that shut down more than one turbine are very rare in practice. Some tasks shut down the turbines when the technicians are effectively working on the task, but not necessarily during the rest time periods they overlap. This is the case for the majority of the preventive maintenance jobs, as well as for wind turbines retrofit. Other tasks do not have any impact on electricity production (e.g., some wind farm inspections). We model the impact on power production of the tasks using two parameters. Parameter b_{wi} takes the value 1 if task $i \in \mathcal{I}$ shuts down turbine $w \in \mathcal{W}$ when technicians are effectively working on the task, and 0 otherwise. Parameter $\widetilde{b}_{wi}$ takes the value of 1 if task i additionally shuts down turbine w during the rest time periods it overlaps, and 0 otherwise. It must be noted that parameters b_{wi} and $\widetilde{b}_{wi}$ are equal to 0 if turbine w is not located at wind farm l_i.

To execute the maintenance tasks, we have a finite set $\mathcal{R}$ of technicians. To avoid time-consuming traveling between distant locations, during a single day technicians can only perform tasks at *compatible locations*. Compatible locations are simply those that can be reached from each other in travel times that are negligible with respect to the duration of a time period in $\mathcal{T}$. Let us assume that t_{max} is the maximum travel time between two locations that we can consider "negligible" with respect to the duration of a time period. The top of Fig. 1 then shows the locations that are compatible with l_1 (i.e., l_2 and l_3). To model these *daily location-based incompatibilities*, we introduce binary parameter $\sigma_{ll'}$ taking the value of 1 if and only if locations l and l' are compatible (naturally $\sigma_{ll'} = \sigma_{l'l}$). The bottom of Fig. 1 shows the 4 sets of compatible locations in our example. During a single day, one should observe that a technician can only execute tasks at l_1

and l_2 or l_3 but not both. It is worth mentioning that wind turbine maintenance tasks usually span along hours (if not days), and therefore, technicians tend to travel between very few locations during a single working day.

We assume that all the technicians work the same shift, which is a common practice in this industry. Nonetheless, each technician $r \in \mathcal{R}$ has an individual availability schedule expressed by a binary vector π_r, with $\pi_r^t = 1$ if r is available during time period $t \in \mathcal{T}$, and $\pi_r^t = 0$ otherwise. The availability schedule of every technician is related to training time, personal holiday time and assignments to tasks (not part of the optimization) that have been already started or that are performed along with external companies. When a technician r is not available during a time period t, his or her location is fixed to $l_r^t \in \mathcal{L}$. Notice that for technician personal holidays and training sessions, this parameter is set to a dummy location l^* such that $\forall l \in \mathcal{L}, \sigma_{l^*l} = 1$.

Technicians master specific skills from a set $\mathcal{S}$. Technician skills are expressed by a binary vector λ_r over $\mathcal{S}$ such that $\lambda_{rs} = 1$ if technician $r \in \mathcal{R}$ masters skill $s \in \mathcal{S}$, and $\lambda_{rs} = 0$ otherwise. Each task $i \in \mathcal{I}$ has a set $\mathcal{M}_i$ of execution modes. For each mode $m \in \mathcal{M}_i$, there is an associated task duration d_{im} and a number q_{im} of required technicians. Switching modes after starting the execution of a task is forbidden. Additionally, only technicians mastering a specific skill $s_i \in \mathcal{S}$ can work on task i. For the sake of clarity, we denote as $\mathcal{R}_i$ the set of technicians that can perform task i. Note that $\mathcal{R}_i = \{r \in \mathcal{R} | \lambda_{rs_i} = 1\}$. We consider that a technician cannot perform more than one task during a given time period. Moreover, a technician assigned to a task has to work on it from the beginning to the end, even if the task is interrupted during one or multiple rest time periods.

Tasks can only be executed during some specific time periods. These take into account maintenance periodicity time windows, spare parts availability, safety work conditions (e.g., a technician cannot perform certain tasks on a turbine when the wind is too strong) and external restrictions imposed by the operator and/or the wind farms owners. To model these restrictions, we introduce parameter γ_i^t that takes the value 1 if task $i \in \mathcal{I}$ can be performed during time period $t \in \mathcal{T}$, 0 otherwise. Additionally, some subsets of tasks cannot overlap due, for instance, to the use of disjunctive resources, an interference (e.g., two tasks cannot be executed on the same turbine at the same time) or managerial preferences. We define $ov(\mathcal{I})$ the set containing all subsets of tasks that should not overlap.

The objective of the problem is to determine a schedule that maximizes the revenue generated by the electricity production of the wind farms while meeting the constraints described above. We denote g_w^t the revenue generated by wind turbine $w \in \mathcal{W}$ if it can produce electricity during time period $t \in \mathcal{T}$. Similarly, we denote $\widetilde{g}_w^d$ the revenue generated by wind turbine w if it can produce electricity during the rest time period following day $d \in \mathcal{D}$. The revenue values are estimated according to the forecasted wind speed. In this study, we do not consider other maintenance costs: We assume that, as it is common in practice, technicians earn a fixed salary, and we disregard travel costs as they are insignificant. One particularity of this problem is the possibility to postpone the scheduling of some tasks until the next planning horizon. To model the postponement of task $i \in \mathcal{I}$, we create an additional execution mode m_i^0 and we add it to $\mathcal{M}_i$ (we have $q_{im_i^0} = 0$ and $d_{im_i^0} = 0$). When task i is postponed, we apply to the objective a penalty of $c_i \geq 0$. In practice, the value of this penalty is fixed according to multiple factors. It takes into account the relative degree of priority of the tasks. This priority depends on reliability consideration (the more a maintenance operation is delayed, the higher is the probability of failure) and contract commitments. Moreover, when a task is postponed, it obviously does not impact the production of any wind turbines and thus the value of the revenue. Therefore, if a task needs to be scheduled during the time horizon, this penalty is fixed in connection to the revenue in order to ensure that the postponement of this task is non-profitable. This penalty includes an estimation of the loss of revenue induced by the schedule of the corresponding task, to which may be added outsourcing costs (the decision maker then being responsible for the choice of outsourcing a task rather than postponing it). Notice that if the penalties are high enough, postponing a task is just triggered to overcome a possible lack of technicians. In short, the objective function to be maximized in the problem always corresponds to the difference between the revenue and the postponing penalties. "Appendix 1" summarizes the notation used in this paper.

3 Integer linear programming formulations

In the following subsections, we present two integer linear programming models for the problem. The first formulation is an immediate *natural* formulation, whereas the second one aggregates some decision variables leading to a more compact formulation.

3.1 Natural formulation

Let us introduce the following decision variables:

$$x_{im} = \begin{cases} 1 & \text{if task } i \in \mathcal{I} \text{ is executed in mode } m \in \mathcal{M}_i, \\ 0 & \text{otherwise.} \end{cases}$$

$$s_i^t = \begin{cases} 1 & \text{if task } i \in \mathcal{I} \text{ starts at the beginning of time period } t \in \mathcal{T}, \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{ri} = \begin{cases} 1 & \text{if technician } r \in \mathcal{R} \text{ is assigned to task } i \in \mathcal{I}, \\ 0 & \text{otherwise.} \end{cases}$$

$$c_i^t = \begin{cases} 1 & \text{if task } i \in \mathcal{I} \text{ ends at the end of time period } t-1 \in \mathcal{T}, \\ 0 & \text{otherwise.} \end{cases}$$

$$e_i^t = \begin{cases} 1 & \text{if task } i \in \mathcal{I} \text{ is executed during time period } t \in \mathcal{T}, \\ 0 & \text{otherwise.} \end{cases}$$

$$u_i^d = \begin{cases} 1 & \text{if task } i \in \mathcal{I} \text{ is executed during day } d \in \mathcal{D}, \\ 0 & \text{otherwise.} \end{cases}$$

$$f_w^t = \begin{cases} 1 & \text{if turbine } w \in \mathcal{W} \text{ can produce electricity} \\ & \text{during time period } t \in \mathcal{T}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\widetilde{f}_w^d = \begin{cases} 1 & \text{if turbine } w \in \mathcal{W} \text{ can produce electricity} \\ & \text{during the rest time period following day } d \in \mathcal{D}, \\ 0 & \text{otherwise.} \end{cases}$$

$$z_{ri}^t = \begin{cases} 1 & \text{if technician } r \in \mathcal{R} \text{ is assigned to task} \\ & i \in \mathcal{I} \text{ during time period } t \in \mathcal{T}, \\ 0 & \text{otherwise.} \end{cases}$$

$$v_{rl}^t = \begin{cases} 1 & \text{if technician } r \in \mathcal{R} \text{ is at location } l \in \mathcal{L} \\ & \text{during time period } t \in \mathcal{T}, \\ 0 & \text{otherwise.} \end{cases}$$

An intuitive formulation is defined as the following integer linear program $[P1]$:

$$[P1] \quad \max \sum_{w \in \mathcal{W}} \left(\sum_{t \in \mathcal{T}} g_w^t f_w^t + \sum_{d \in \mathcal{D}} \widetilde{g}_w^d \widetilde{f}_w^d \right) - \sum_{i \in \mathcal{I}} o_i x_{im_i^0} \tag{1}$$

subject to:

$$\sum_{m \in \mathcal{M}_i} x_{im} = 1 \quad \forall i \in \mathcal{I}, \tag{2}$$

$$e_i^0 = 0 \quad \forall i \in \mathcal{I}, \tag{3}$$

$$e_i^t = e_i^{t-1} + s_i^t - c_i^t \quad \forall i \in \mathcal{I}, \quad \forall t \in \mathcal{T} \setminus \{0\}, \tag{4}$$

$$\sum_{t \in \mathcal{T}} s_i^t = 1 \quad \forall i \in \mathcal{I}, \tag{5}$$

$$\sum_{t \in \mathcal{T}} c_i^t = 1 \quad \forall i \in \mathcal{I}, \tag{6}$$

$$e_i^t \leq \gamma_i^t \quad \forall i \in \mathcal{I}, \quad \forall t \in \mathcal{T}, \tag{7}$$

$$\sum_{i \in \mathcal{B}} e_i^t \leq 1 \quad \forall \mathcal{B} \in ov(\mathcal{I}), \quad \forall t \in \mathcal{T}, \tag{8}$$

$$\sum_{t \in \mathcal{T}_d} e_i^t \leq |\mathcal{T}_d| u_i^d \quad \forall i \in \mathcal{I}, \quad \forall d \in \mathcal{D}, \tag{9}$$

$$f_w^t + b_{wi} e_i^t \leq 1 \quad \forall w \in \mathcal{W}, \quad \forall i \in \mathcal{I}, \quad \forall t \in \mathcal{T}, \tag{10}$$

$$\widetilde{f}_w^d + \widetilde{b}_{wi} \left(u_i^d + u_i^{d+1} \right) \leq 2 \quad \forall w \in \mathcal{W}, \quad \forall i \in \mathcal{I}, \quad \forall d \in \mathcal{D}, \tag{11}$$

$$\sum_{t \in \mathcal{T}} e_i^t = \sum_{m \in \mathcal{M}_i} d_{im} x_{im} \quad \forall i \in \mathcal{I}, \tag{12}$$

$$\sum_{r \in \mathcal{R}_i} y_{ri} = \sum_{m \in \mathcal{M}_i} q_{im} x_{im} \quad \forall i \in \mathcal{I}, \tag{13}$$

$$e_i^t + y_{ri} - z_{ri}^t \leq 1 \quad \forall i \in \mathcal{I}, \quad \forall r \in \mathcal{R}_i, \quad \forall t \in \mathcal{T}, \tag{14}$$

$$z_{ri}^t \leq y_{ri} \quad \forall i \in \mathcal{I}, \quad \forall r \in \mathcal{R}_i, \quad \forall t \in \mathcal{T}, \tag{15}$$

$$z_{ri}^t \leq e_i^t \quad \forall i \in \mathcal{I}, \quad \forall r \in \mathcal{R}_i, \quad \forall t \in \mathcal{T}, \tag{16}$$

$$\sum_{i \in \mathcal{I}_l \cap \mathcal{R}_i} z_{ri}^t \leq \pi_r^t v_{rl}^t \quad \forall r \in \mathcal{R}, \quad \forall l \in \mathcal{L}, \quad \forall t \in \mathcal{T}, \tag{17}$$

$$\sum_{l \in \mathcal{L}} v_{rl}^t = 1 \quad \forall r \in \mathcal{R}, \quad \forall t \in \mathcal{T}, \tag{18}$$

$$v_{rl_r^t}^t = 1 \quad \forall r \in \mathcal{R}, \quad \forall t \in \mathcal{T} \text{ s.t.} \pi_r^t = 0, \tag{19}$$

$$v_{rl}^t + \sum_{l' \in \mathcal{L} | \sigma_{ll'} = 0} v_{rl'}^{t'} \leq 1 \quad \forall r \in \mathcal{R}, \quad \forall d \in \mathcal{D}, \quad \forall (t, t') \in \mathcal{T}_d \times \mathcal{T}_d, t \neq t', \quad \forall l \in \mathcal{L}, \tag{20}$$

$$e_i^t, s_i^t, c_i^t \in \{0, 1\} \quad \forall i \in \mathcal{I}, \quad \forall t \in \mathcal{T}, \tag{21}$$

$$u_i^d \in \{0, 1\} \quad \forall i \in \mathcal{I}, \quad \forall d \in \mathcal{D}, \tag{22}$$

$$f_w^t \in \{0, 1\} \quad \forall w \in \mathcal{W}, \forall t \in \mathcal{T}, \tag{23}$$

$$\widetilde{f}_w^d \in \{0, 1\} \quad \forall w \in \mathcal{W}, \quad \forall d \in \mathcal{D}, \tag{24}$$

$$y_{ri} \in \{0, 1\} \quad \forall i \in \mathcal{I}, \quad \forall r \in \mathcal{R}_i, \tag{25}$$

$$z_{ri}^t \in \{0, 1\} \quad \forall i \in \mathcal{I}, \quad \forall r \in \mathcal{R}_i, \quad \forall t \in \mathcal{T}, \tag{26}$$

$$v_{rl}^t \in \{0, 1\} \quad \forall r \in \mathcal{R} \quad , \forall l \in \mathcal{L}, \quad \forall t \in \mathcal{T}. \tag{27}$$

The objective in (1) is defined as the difference between the revenue generated by the turbines and the penalties induced by the postponement of some tasks. Constraints (2) guarantee that exactly one execution mode is selected for each task. For each task, constraints (3)–(6) ensure consistency between the starting time, ending time, and execution time period variables. Constraints (7) prevent a task to be scheduled during forbidden time periods. Constraints (8) are the non-overlapping constraints. Constraints (9) couple the time periods during which each task is performed to the execution days of this task. Constraints (10) and (11) compute the impact of the tasks on the availability of the turbines to produce electricity. Constraints (12) connect the duration of each task to its selected execution mode. Constraints (13) ensure that the technician requirements are fulfilled. Constraints (14) force technicians to be assigned to a task from its beginning to its end. For each technician, constraints (15)–(16) ensure consistency between the global assignment and the time-indexed assignment variables. Constraints (17) couple the locations of the technicians to the tasks they perform. Constraints (18) prevent technicians to perform multiple tasks during the same time period. When technicians are not available, constraints (19) ensure compliance with their known locations. Constraints (20) define the daily location-based incompatibilities for each technician. Finally, (21)–(27) state the binary nature of the decision variables.

3.2 Compact formulation

In order to restrict the number of constraints involved in the formulation $[P1]$, we propose a second model based on the concept of *plans*. A plan associated with task $i \in \mathcal{I}$ defines

a feasible schedule for i by setting an execution mode, a consistent starting date, and, by induction, a duration and a resource requirement. For example, consider a task i with two execution modes m_1 and m_2. Let d_{m_1} and d_{m_2} denote the corresponding durations and q_{m_1} and q_{m_2} the corresponding number of required technicians. Assume that task i can be executed during the whole time horizon. For each $t \in \mathcal{T}$ such that $t \leq |T| - d_{m_1}$, a feasible plan is created to represent the planning of task i within mode m_1 from period t to period $t + d_{m_1}$ with a requirement of q_{m_1} technicians. The same procedure is applied for mode m_2. Obviously, we take into consideration the impossibility of preempting tasks when building plans.

All the plans are generated a priori. Since in practice the planning horizon is short (because of weather predictions) and there are only a few execution modes, the total number of plans is not so large. We denote by $\mathcal{P}$ the set of plans, i_p the task involved in plan $p \in \mathcal{P}$, and $\mathcal{P}_i$ the set of all plans involving task i (i.e., $\mathcal{P}_i = \{p \in \mathcal{P} | i_p = i\}$). For each task i, we also create a plan $p_i^0 \in \mathcal{P}_i$ representing the postponement of the task. For a plan p, execution periods of i_p are expressed by a binary vector a_p where $a_p^t = 1$ if i_p is executed during time period $t \in \mathcal{T}$, and $a_p^t = 0$ otherwise. Similarly, we introduce binary vector $\widetilde{a}_p$ where $\widetilde{a}_p^d = 1$ if i_p overlaps the rest time period following day $d \in \mathcal{D}$, and $\widetilde{a}_p^d = 0$ otherwise. With a slight abuse of notation, we introduce parameters b_{wp}, $\widetilde{b}_{wp}$, and $\mathcal{R}_p$, respectively, equal to b_{wi_p}, $\widetilde{b}_{wi_p}$ and $\mathcal{R}_{i_p}$. Moreover, we define q_p as the number of technicians required when selecting plan $p \in \mathcal{P}$. Finally, parameter o_p is the penalty if plan p is selected (note that $\forall i \in \mathcal{I}, \forall p \in \mathcal{P}_i \setminus \{p_i^0\},\ o_p = 0$ and $o_{p_i^0} = o_i$).

Scheduling the tasks becomes rather implicit as it simply requires to select a plan for each task. Nevertheless, we still need to manage the technician-to-task assignments that should meet the daily location-based incompatibilities and cope with technician availability. We use the decision variables f_w^t, $\widetilde{f}_w^d$, and v_{rl}^t defined in Sect. 3.1. We also introduce the following decision variables:

$$\bar{x}_p = \begin{cases} 1 & \text{if plan } p \in \mathcal{P} \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

$$\bar{y}_{rp} = \begin{cases} 1 & \text{if technician } r \in \mathcal{R}_p \text{ is assigned to plan } p \in \mathcal{P}, \\ 0 & \text{otherwise.} \end{cases}$$

As a result, we obtain the following integer linear program, denoted as $[P2]$.

$$[P2] \quad \max \sum_{w \in \mathcal{W}} \left(\sum_{t \in \mathcal{T}} g_w^t f_w^t + \sum_{d \in \mathcal{D}} \widetilde{g}_w^d \widetilde{f}_w^d \right) - \sum_{p \in \mathcal{P}} o_p \bar{x}_p \tag{28}$$

subject to:

$$\sum_{p \in \mathcal{P}_i} \bar{x}_p = 1 \quad \forall i \in \mathcal{I}, \tag{29}$$

$$\sum_{i \in \mathcal{B}} \sum_{p \in \mathcal{P}_i} a_p^t \bar{x}_p \leq 1 \quad \forall \mathcal{B} \in ov(\mathcal{I}), \quad \forall t \in \mathcal{T}, \tag{30}$$

$$f_w^t + \sum_{p \in \mathcal{P}_i} b_{wp} a_p^t \bar{x}_p \leq 1 \quad \forall w \in \mathcal{W}, \quad \forall i \in \mathcal{I}, \quad \forall t \in \mathcal{T}, \tag{31}$$

$$\widetilde{f}_w^d + \sum_{p \in \mathcal{P}_i} \widetilde{b}_{wp} \widetilde{a}_p^d \bar{x}_p \leq 1 \quad \forall w \in \mathcal{W}, \forall i \in \mathcal{I}, \quad \forall d \in \mathcal{D}, \tag{32}$$

$$\sum_{r \in \mathcal{R}_p} \bar{y}_{rp} = q_p \bar{x}_p \quad \forall p \in \mathcal{P}, \tag{33}$$

$$\sum_{i \in \mathcal{I}_l | r \in \mathcal{R}_i} \sum_{p \in \mathcal{P}_i} a_p^t \bar{y}_{rp} \leq \pi_r^t v_{rl}^t \quad \forall r \in \mathcal{R}, \quad \forall l \in \mathcal{L}, \quad \forall t \in \mathcal{T}, \tag{34}$$

$$(18), (19), (20),$$

$$\bar{x}_p \in \{0, 1\} \quad \forall p \in \mathcal{P}, \tag{35}$$

$$\bar{y}_{rp} \in \{0, 1\} \quad \forall p \in \mathcal{P}, \quad \forall r \in \mathcal{R}_p,$$

$$(23), (24), (27). \tag{36}$$

The objective in (28) is defined as the difference between the revenue generated by the turbines and the penalties induced by the postponement of some tasks. Constraints (29) ensure that exactly one plan is selected for each task. Constraints (30) are the non-overlapping constraints. Constraints (31) and (32) couple turbine availability variables to plan selection variables. Constraints (33) ensure that the technician requirements are fulfilled. Constraints (34) couple the locations of the technicians to the tasks they perform. This new formulation $[P2]$ uses the same constraints as formulation $[P1]$ to manage the availability calendars of the technicians and the daily location-based incompatibilities. Finally, (35)–(36) state the binary nature of the new decision variables.

4 Constraint programming formulation

The previous section presents two ILP formulations of the problem. Motivated by the successful implementation of CP models for solving other hard, and to some extent, related optimization problems (Baptiste et al. 2001; Rodriguez 2007; Malapert et al. 2012), we also decided to approach our problem using CP.

First of all, note that defining for each task: i) an execution mode, ii) a starting time and iii) the technicians assigned to it, is enough to obtain a solution to our problem. Therefore, for each task $i \in \mathcal{I}$, we introduce the variables $M_i \in \mathcal{M}_i$ and $S_i \in \mathcal{T}$ to represent its execution mode and starting time period, and we use the binary variables $(y_{ri})_{r \in \mathcal{R}_i}$ introduced in Sect. 3.1 to decide if technician r performs or not task i. To make some constraints easier to model, we introduce integer

variables $C_i \in \mathcal{T}$, $D_i \in \{d_{im}\}_{m \in \mathcal{M}_i}$, $Q_i \in \{q_{im}\}_{m \in \mathcal{M}_i}$ and set variables $E_i \subseteq \mathcal{T}$ defining for task i its completion time period, its duration, its number of assigned technicians and its set of execution time periods, respectively.

Execution time periods of each task are coupled to their starting and ending time periods with constraints (37)–(38).

$$S_i + D_i - 1 = C_i \quad \forall i \in \mathcal{I}, \tag{37}$$

$$t \in E_i \Leftrightarrow t \in [S_i, C_i] \cap \mathbb{N} \quad \forall i \in \mathcal{I} \tag{38}$$

The duration of each task (39) as well as the number of assigned technicians (40) are coupled with the selected execution mode.

$$D_i = d_{iM_i} \quad \forall i \in \mathcal{I}, \tag{39}$$

$$Q_i = q_{iM_i} \quad \forall i \in \mathcal{I} \tag{40}$$

Constraints (41) are the non-overlapping constraints.

$$\bigcap_{i \in \mathcal{B}} E_i = \emptyset \quad \forall \mathcal{B} \in ov(\mathcal{I}) \tag{41}$$

Constraints (42) ensure that the technician requirements are fulfilled for each task.

$$\sum_{r \in \mathcal{R}_i} y_{ri} = Q_i \quad \forall i \in \mathcal{I} \tag{42}$$

To express the constraints related to the technician-to-task assignments, we introduce set variables $Y_r^t \subseteq \mathcal{I} \cup \{i^0\}$ defining the set of tasks that technician r could potentially perform during time period $t \in \mathcal{T}$. Index i^0 represents a dummy task, created in order to prevent a technician to work when he or she is unavailable. Constraints (43) couple these variables to the global assignment variables $(y_{ri})_{i \in \mathcal{I} | r \in \mathcal{R}_i}$. Restrictions imposed on the locations visited by a technician within each day lead to the introduction of set variables $V_r^t \subseteq \mathcal{L}$ defining the set of potential locations for technician r during time period t. Constraints (44) and (45) restrict the set of tasks that a technician can possibly execute according to his or her potential locations. Set $\mathcal{L}(\hat{\mathcal{I}})$ defines the set of locations of the tasks in set $\hat{\mathcal{I}}$. Note that $\mathcal{L}(\hat{\mathcal{I}}) = \{l \in \mathcal{L} \mid \exists i \in \hat{\mathcal{I}} \text{ s.t. } l_i = l\}$.

$$y_{ri} = 1 \Rightarrow \left(Y_r^t = \{i\} \quad \forall t \in E_i\right) \quad \forall i \in \mathcal{I}, \quad \forall r \in \mathcal{R}_i, \tag{43}$$

$$V_r^t = \mathcal{L}(Y_r^t) \quad \forall r \in \mathcal{R}, \quad \forall t \in \mathcal{T}, \text{ s.t. } \pi_r^t = 1, \tag{44}$$

$$V_r^t = \{l_r^t\} \wedge Y_r^t = \{i^0\} \quad \forall r \in \mathcal{R}, \quad \forall t \in \mathcal{T}, \text{ s.t. } \pi_r^t = 0 \tag{45}$$

Denoting d_t as the day to which time period t belongs, constraints (46) ensure that the daily location-based incompatibilites are not violated for each technician.

$$V_r^t = \{l\} \Rightarrow \left(l' \notin V_r^{t'} \quad \forall l' \in \mathcal{L} \text{ s.t. } \sigma_{ll'} = 0, \quad \forall t' \in \mathcal{T}_{d_t} \text{ s.t. } t' \neq t\right) \quad \forall r \in \mathcal{R}, \quad \forall t \in \mathcal{T}, \quad \forall l \in \mathcal{L} \tag{46}$$

In order to define the objective function of our problem, we introduce two set variables. Variables $F_w^{day} \subseteq \{1, \ldots, |\mathcal{T}|\}$ define the set of all periods during which turbine $w \in \mathcal{W}$ can produce electricity. Variables $F_w^{rest} \subseteq \{1, \ldots, |\mathcal{D}|\}$ define the set of days for which turbine w can produce electricity during the corresponding rest time periods. More specifically, a day d belongs to this set if w can produce electricity during the rest time period between d and $d+1$. Additionally, we denote by t_d^{rest} the last time period $t \in \mathcal{T}$ before the rest time period following day $d \in \mathcal{D}$.

We introduce constraints (47), (48), (49) and (50) which state that a turbine is available to produce electricity during a time period if and only if no tasks requiring its shutdown are scheduled during this period.

$$t \in E_i \Rightarrow t \notin F_w^{day} \quad \forall w \in \mathcal{W}, \quad \forall i \in \mathcal{I} \text{ s.t. } b_{wi} = 1, \quad \forall t \in \mathcal{T}, \tag{47}$$

$$t \notin \bigcup_{i \in \mathcal{I} | b_{wi} = 1} E_i \Rightarrow t \in F_w^{day} \quad \forall w \in \mathcal{W}, \quad \forall t \in \mathcal{T}, \tag{48}$$

$$t_d^{rest} \in E_i \wedge (t_d^{rest} + 1) \in E_i \Rightarrow d \notin F_w^{rest} \quad \forall w \in \mathcal{W}, \quad \forall i \in \mathcal{I}, \text{ s.t. } \widetilde{b}_{wi} = 1, \quad \forall d \in \mathcal{D}, \tag{49}$$

$$\bigwedge_{i \in \mathcal{I} | \widetilde{b}_{wi} = 1} \left(\{t_d^{rest}, t_d^{rest} + 1\} \nsubseteq E_i\right) \Rightarrow d \in F_w^{rest} \quad \forall w \in \mathcal{W}, \quad \forall d \in \mathcal{D} \tag{50}$$

Constraint (51) defines the objective function variable $obj \in \mathbb{R}$ of our problem.

$$obj = \sum_{w \in \mathcal{W}} \left(\sum_{t \in F_w^{day}} g_w^t + \sum_{d \in F_w^{rest}} \widetilde{g}_w^d \right) - \sum_{i \in \mathcal{I} | M_i = m_i^0} o_i \tag{51}$$

To remove some symmetries, we add constraints (52) to impose the starting time of a postponed task to be equal to 0.

$$M_i = m_i^0 \Leftrightarrow S_i = 0 \qquad \forall i \in \mathcal{I} \tag{52}$$

5 A CP-based large neighborhood approach

We use the CP model introduced in Sect. 4 as the main building block of a CP-based large neighborhood search (CPLNS) approach.

This method is based on the *large neighborhood search* metaheuristic (LNS) originally proposed by Shaw (1998) for a vehicle routing problem. In the LNS, the current solution is successively partially destroyed and repaired in order to

improve its quality. Our implementation randomly selects the operators with equal probability as suggested in Pisinger and Ropke (2010).[4]

Algorithm 1 outlines the general structure of the method. To compute the initial solution, we use the CP model and we stop its execution as soon as we find a feasible solution. The algorithm then enters an iterative process. In every iteration, it randomly selects a destroy operator o_1 and a repair operator o_2. First, it partially destroys the current solution using o_1 (see Sect. 5.1). Then, it builds a potential alternative solution sol' using o_2 (see Sect. 5.2). In the case where sol' meets the acceptance criterion (see Sect. 5.3), the solution sol' replaces the current solution sol for the next iteration. If appropriate, the algorithm updates the best solution sol^* found so far. Then, the search moves to the next iteration. The whole procedure is repeated until it reaches a time limit. The optimization returns solution sol^*.

Algorithm 1: Script of the *CPLNS* algorithm

```
1  sol ← initial solution
2  sol* ← sol
3  repeat
4  |  Select a destroy operator o1 and a repair operator o2 from the
   |  operators pool
5  |  sol' ← repair (o2, destroy (o1, s), sol)
6  |  if sol' is accepted then
7  |  |  sol ← sol'
8  |  end
9  |  if f(sol') > f(sol*) then
10 |  |  sol* ← sol'
11 |  end
12 until the time limit is reached;
13 return sol*
```

5.1 Destroy operators

At each iteration, the algorithm selects Γ tasks to remove from the current solution. The value of Γ is randomly fixed in the interval $\left[\max\left(n^-, n \times p^-\right); \min\left(n^+, n \times p^+\right)\right]$, where n^- and n^+ denote the minimal and maximal number of tasks that are allowed to be removed during an iteration; similarly, p^- and p^+ denote the minimal and maximal proportion of tasks that could be removed. The parameters p^- and p^+ allow the algorithm to adapt to all instances independently of their size. We use the following settings: $\left(n^-, n^+, p^-, p^+\right) = (5, 20, 0.1, 0.4)$. We also always consider postponed tasks in the current solution as tasks to be removed. However, we do not count them among the Γ tasks to remove.

After setting Γ, the algorithm selects the tasks using one of the following six removal operators:

- *Operator A: random removal*
  This operator randomly removes Γ tasks from the current solution. The intention behind this operator is to diversify the search.
- *Operator B: worst removal*
  This operator removes the tasks which penalize the most the objective function of the current solution. Let f be the current value of the objective function, f_{-i} its value if task i is removed, and $\Delta f(i) = f - f_{-i}$. The Γ tasks with the greatest values of $\Delta f(i)$ are removed from the current solution in order to insert them at better positions.
- *Operator C: technicians duties removal*
  This operator is based on the following procedure. First, it randomly selects a skill s^*. Second, as long as the number of removed tasks is lower than Γ, it randomly selects a technician mastering s^* and remove from the current solution those tasks in which the selected technician uses skill s^*. The operator then switches to another skill if it has not removed Γ tasks yet. Freeing up a pool of technicians along the whole time horizon may allow the reinsertion of possibly misplaced tasks during more convenient time periods (i.e, periods where they penalize less the revenue).
- *Operator D: similar tasks removal*
  This operator removes *similar tasks*. More specifically, the operator aims to remove non-overlapping tasks (or tasks that overlap as little as possible) having similar duration and skill requirements. The similarity between two tasks $i, j \in \mathcal{I}$ in a solution sol is formally defined as: $\phi(i, j, sol) = \alpha_1 \times |\bar{d}_i - \bar{d}_j| + \alpha_2 \times \mathbb{1}^5_{(s_i \neq s_j)} + \alpha_3 \times ov(i, j, sol)$, where $\bar{d}_i$ is the average duration of task i (i.e., $\bar{d}_i = \frac{1}{|\mathcal{M}_i \setminus \{m_i^0\}|} \sum_{m \in \mathcal{M}_i \setminus \{m_i^0\}} d_{im}$) and symbol $\mathbb{1}_{(s_i \neq s_j)}$ is equal to 1 if $s_i \neq s_j$, 0 otherwise. Function $ov(i, j, sol)$ computes the number of overlapping time periods between i and j in the current solution sol. Coefficients α_1, α_2 and α_3 weight the three components of the similarity function, namely task duration, skill requirements and task overlapping. In our experiments, $(\alpha_1, \alpha_2, \alpha_3) = (1, 3, 5)$. To select the tasks to remove, the operator first initializes a set $\widetilde{\mathcal{I}}$ with a random task. While $|\widetilde{\mathcal{I}}| \leq \Gamma$, the procedure randomly selects a task i^* from $\widetilde{\mathcal{I}}$, and it then adds to $\widetilde{\mathcal{I}}$ the task $j \in \mathcal{I} \setminus \widetilde{\mathcal{I}}$ with the minimal value of $\phi(i^*, j, sol)$. The intuition behind this opera-

[4] We also implemented the adaptive layer as proposed in Ropke and Pisinger (2006), but after some preliminary experimentation we concluded that the contribution of this component to the accuracy of the method did not payoff the loss of simplicity and the effort needed to fine tune the additional parameters. We therefore limit the discussion in the paper to the basic LNS version.

tor is that removing and reinserting similar tasks that are scheduled in non-overlapping time periods increases the likelihood of a solution improvement.

– *Operator E: task maximal regret*
This operator removes the tasks having the largest difference between the loss of revenue they currently generate and the minimal loss of revenue they can induce (we called this difference *regret*). Let $\mathcal{W}_i$ denote the set of turbines shut down by the execution of a task i (clearly, $\mathcal{W}_i = \{w \in \mathcal{W} | b_{wi} = 1 \vee \widetilde{b}_{wi} = 1\}$). The loss induced by task i is equal to the sum over all the turbines in $\mathcal{W}_i$ of the revenue directly lost due to its scheduling. Notice that if multiple tasks impact a turbine during a specific time period, the loss is set proportionally to the number of these tasks. Prior to the optimization, the operator computes for each task i a metric called $loss_i^{best}$ equal to the smallest loss of revenue that can be achieved when one only considers the scheduling of this task. Then, during the optimization, the operator first computes the loss of revenue $loss_i^{sol}$ generated by task i in the current solution sol. Afterward, the operator computes the regret $\Delta loss(i) = loss_i^{sol} - loss_i^{best}$ for each scheduled task i. The operator then removes from the current solution sol the Γ tasks associated with the largest value of $\Delta loss(i)$. Removing tasks that currently generate considerably more loss of revenue than they could may allow the algorithm to schedule those tasks in better positions in the next iterations. It is then plausible to assume that this operator increases the probability of fining better quality solutions.

– *Operator F: turbine maximal regret*
This operator works almost in the same way as operator E. Instead of reasoning by task, we focus on each turbine. Prior to the optimization, the procedure computes for each turbine $w \in \mathcal{W}$ a metric called $loss_w^{best}$, estimating the smallest loss of revenue that can be achieved when one only considers the set $\mathcal{I}_w$ of tasks that prevent turbine w to produce electricity when scheduled (i.e., $\mathcal{I}_w = \{i \in \mathcal{I} | b_{wi} = 1 \vee \widetilde{b}_{wi} = 1\}$). The value of $loss_w^{best}$ is computed by running the CP formulation presented in Sect. 4 on an instance containing only the tasks belonging to $\mathcal{I}_w$. The solution time is most of the time insignificant, but nevertheless we impose a time limit of 1 s. It is noteworthy that, if we find a smaller loss of revenue during the execution of the CPLNS, we update the value of $loss_w^{best}$. Our tests, however, suggest that this is a very rare event. During the optimization, the procedure starts by computing the lost revenue $loss_w^{sol}$ generated by the tasks in $\mathcal{I}_w$ if they are executed as scheduled in the current solution. Notice that the penalties related to postponed tasks are included in the computation of $loss_w^{best}$ and $loss_w^{sol}$. Afterward, the operator initializes a set $\widetilde{\mathcal{W}}$ with all the turbines in $\mathcal{W}$ and compute the regret $\Delta loss(w) = loss_w^{sol} - loss_w^{best}$ associated with each turbine $w \in \widetilde{\mathcal{W}}$. While $\widetilde{\mathcal{W}}$ is not empty and Γ tasks are not removed, the operator removes from $\widetilde{\mathcal{W}}$ the turbine w^* associated with the largest value of $\Delta loss(w)$ and removes from the current solution sol all the scheduled tasks belonging to $\mathcal{I}_{w^*}$.

We work with randomized versions of operators B, D, E and F to explore the search space more broadly. Indeed, an operator can destroy different parts of the same solution each time it is applied to it. This can then lead to building different solutions. Although the randomization strategy we use is relatively simple, we explain it here for the sake of completeness. The strategy is based on the one proposed in Cordeau et al. (2010). Let ϱ_o denote the *randomization factor* of operator o. When selecting tasks for removal, the operator first sorts a list L containing all the tasks using its selection criterion (i.e., largest penalization for operator B, largest similarity with a specified task for operator D, largest regret for operators E and F). The first positions of L contains the tasks that the destroy operator has to target first according to its criterion. Then the operator draws a random number $y \in [0; 1)$ and it selects for removal task i in position $\lfloor y^{\varrho_o} \times |L| \rfloor$ in L (positions in L are indexed from 0). A randomization factor $\varrho_o = 1$ makes the operator completely random, while higher values of ϱ_o make the operators more deterministic. In our experiments we set $\rho_B = \rho_D = \rho_E = \rho_F = 3$ and we use only the randomized versions of these four operators.

Although it is very simple, Algorithm 2 presents the general structure of a destroy operator used as a subroutine in Algorithm 1.

Algorithm 2: Destroy(o,sol)

Data: a solution sol
a destroy operator o
Result: a set of tasks to remove from sol
1 $\mathcal{F} \leftarrow \emptyset$
2 $\mathcal{F} \leftarrow$Apply destroy operator o to sol
3 **return** $\mathcal{F}$

5.2 Repair operators

We use the CP formulation introduced in Sect. 4 to repair partially destroyed solutions. More specifically, if $\mathcal{F}$ denotes the set of tasks that have been removed, we fix for each task $i \in \mathcal{I} \setminus \mathcal{F}$ the value of the variables M_i, S_i, and $(y_{ri})_{r \in \mathcal{R}_i}$ to their value in the current solution, and we solve the resulting model.

A solution to the CP model is found as soon as the decision variables M_i, S_i, and $(y_{ri})_{r \in \mathcal{R}_i}$ are instantiated for every task

$i \in \mathcal{I}$. Therefore, the branching strategy should focus only on these variables. It is worth noting that a CP solver can make meaningful deductions for a task when the domain of the variable related to its executing mode not longer contains the postponement mode. Moreover, fixing the starting time period of a task before knowing its execution mode leads to a weak propagation on the bound of the revenue variable and on the possible starting time periods and execution modes of other tasks. Furthermore, since variables y_{ri} have an impact only on the feasibility of a solution but not on its quality, fixing last these variables (i.e., after having fixed the variables M_i and S_i for each task $i \in \mathcal{I}$) implies that the solver has to explore a large subtree before reconsidering a bad decision. Based on these observations, we adopt a task-by-task scheduling strategy in which the technicians assignment is made after having chosen an execution mode and a starting time period for the current task.

It is well known that quickly reaching a good quality solution increases the efficiency of the search. It is, however, not clear whether fixing the execution mode of a task $i \in \mathcal{I}$ (i.e., $\mathcal{M}_i$) to a specific execution mode and then exploring all its potential starting time periods before setting M_i to another value is the best searching strategy. This observation suggests that simultaneously setting variables M_i and S_i may lead to achieve a greater flexibility during the search. We therefore choose to reuse the notion of plans introduced in Sect. 3.2. For each task $i \in \mathcal{I}$, we introduce variable $X_i \in \mathcal{P}_i$ that defines the plan selected for task i. We add the constraints (53)–(54) to couple these variables to the variables M_i and S_i.

$$M_i = mode_{X_i} \quad \forall i \in \mathcal{I}, \tag{53}$$

$$S_i = start_{X_i} \quad \forall i \in \mathcal{I} \tag{54}$$

For a plan $p \in \mathcal{P}$, $mode_p$ is the selected execution mode for the task i_p and $start_p = \min_{t \in \mathcal{T} | a_p^t = 1} t$ represents the starting time of i_p. In summary, task by task, we first define its execution mode along with its starting time by fixing variable X_i, and we finally assign the required technicians by fixing the variables $(y_{ri})_{r \in \mathcal{R}_i}$.

To reach feasible solutions faster, we maintain arc consistency on constraints (53) and (54). We also designed customized propagators to try to keep, during the search, the domain of X_i consistent with the availability of the technicians. More specifically, these propagators rely on a comparison between the task requirements and the number of technicians available during each time period of the planning horizon considering the required skills and the daily location-based incompatibilities. They also take into account that technicians have to work on a task from its beginning to its end. For instance, if during a time period t^* no more than 2 technicians mastering a specific skill s^* are available, then for each task i such that $s_i = s^*$ we can remove from the domain of X_i all the plans overlapping t^* and requiring more than 2 technicians.

The most critical part of the procedure is the selection of the next task to be considered by the branching strategy. We select the next task to schedule using a look-ahead regret heuristic that operates as follows. Let $\mathcal{I}^0$ denote the set of tasks which have not yet been processed at the current node of the search. We denote Δf_i^k the k-th smallest value of the loss of revenue that task i can generate when scheduled using one of its possible plans. The procedure *regret-q* chooses task $i^* = \arg\max_{i \in \mathcal{I}^0} \sum_{k=2}^{k=q} \left(\Delta f_i^k - \Delta f_i^1\right)$ to be considered for scheduling. The algorithm computes Δf_i^k according to the values of $\Psi(i, p)$, a function representing the loss of revenue if task i uses plan $p \in \mathcal{P}_i$ (i.e., the task is performed in mode $mode_p$ and starts at the beginning of time period $start_p$). Function $\Psi(i, p)$ is computed using functions $\Psi^{day}(i, p)$ and $\Psi^{rest}(i, p)$ which represent, if task i uses plan $p \in \mathcal{P}_i$, the loss of revenue during the time periods from $\mathcal{T}$ and during the rest time periods. These functions are defined as follows:

$$\Psi(i, p) = \Psi^{day}(i, p) + \Psi^{rest}(i, p),$$

$$\Psi^{day}(i, p) = \begin{cases} o_p & \text{if } p = p_i^0, \\ \sum_{w \in \mathcal{W} | b_{wi} = 1} \sum_{t = start_p}^{t < start_p + d_{imode_p}} g(w, t) & \text{otherwise.} \end{cases},$$

$$\Psi^{rest}(i, p) = \begin{cases} 0 & \text{if } p = p_i^0, \\ \sum_{w \in \mathcal{W} | \widetilde{b}_{wi} = 1} \sum_{d \in \mathcal{D}_p} \widetilde{g}(w, d) & \text{otherwise.} \end{cases},$$

where $\mathcal{D}_p$ is the set of days that task i_p of plan $p \in \mathcal{P}$ overlaps. Functions $g(w, t)$ and $\widetilde{g}(w, d)$ are defined as:

$$\forall w \in \mathcal{W}, \quad \forall t \in \mathcal{T}, \quad g(w, t) = \begin{cases} g_w^t & \text{if } t \in Env(F_w^{day}), \\ 0 & \text{otherwise.} \end{cases},$$

$$\forall w \in \mathcal{W}, \quad \forall d \in \mathcal{D}, \quad \widetilde{g}(w, d) = \begin{cases} \widetilde{g}_w^d & \text{if } d \in Env(F_w^{rest}), \\ 0 & \text{otherwise.} \end{cases},$$

where $Env(Z)$ denotes the set of elements that may belong to the set variable Z in a solution at the current node of the search tree.

Let $Dom(z)$ denote the domain of variable z (i.e., all the possible values that z can take). We have $\Delta f_i^1 = \min_{p \in Dom(X_i)} \Psi(i, p)$. More generally, Δf_i^k is the k-th smallest value of $\Psi(i, p)$. Once task i^* has been selected, it is scheduled using plan $p^* = \arg\min_{p \in Dom(X_{i^*})} \Psi(i^*, p)$.

During our preliminary experiments, we observed that sometimes our regret-q heuristic is unable to lead the search to good solutions. It is indeed possible that a task with a small

regret at a given point of the search is not chosen to be scheduled, but that this decision leads to a large loss of revenue later when exploring the associated subtree. To overcome this potential issue, we designed another branching strategy that selects the task $i^* = \arg\max_{i \in \mathcal{I}_0} \left(\min_{p \in Dom(X_i)} \Psi(i, p) \right)$ for which the minimal loss of revenue is maximal. Again, once task i^* has been selected, it is scheduled using plan $p^* = \arg\min_{p \in Dom(X_{i^*})} \Psi(i^*, p)$. We refer to this branching strategy as $MinMaxLoss$.

The resources assignment is then done technician by technician as long as the request is not fulfilled. We choose with priority the compatible technician which is already working during the days that belong to $\mathcal{D}_{p^*}$. Since the daily location-based incompatibilities are very restrictive, it should be preferable to use technicians that are already working at the same location or at compatible locations. Otherwise, the number of technicians that will be available for other tasks, especially those at incompatible locations, may be drastically restricted. If during the days $d \in \mathcal{D}_{p^*}$ multiple technicians work the same number of time periods, we choose first the technician that could perform the least number of tasks among those remaining. If several technicians can still be selected, we select one randomly.

Exploring the whole neighborhood of a solution is time-consuming; therefore, we only allow a certain number ϖ_{max} of backtracks (we set $\varpi_{max} = 200$ in our experiments). Thus, different solutions can be obtained using different branching strategies. Different repair operators are therefore defined using different branching strategies. In our experiments, we use regret-2 and regret-3 branching strategies, as well as a randomized version of $MaxMinLoss$, where the probability of selecting a task is inversely proportional to the minimal loss of revenue it generates at this point of the search.

Algorithm 3 presents the general structure of a repair operator used as a subroutine in Algorithm 1.

Algorithm 3: Repair(o,$\mathcal{F}$,s)

Data: a solution sol
a set $\mathcal{F}$ of tasks
a repair operator o (branching strategy)
Result: a new solution sol'
foreach $i \in \mathcal{I} \setminus \mathcal{F}$ **do**
  Fix the values of M_i, S_i, $(y_{ri})_{r \in \mathcal{R}_i}$ as in solution sol in the CP model
end
Solve the CP model applying repair operator o, yelding sol'
return sol'

5.3 Acceptance criteria

The original version of LNS proposed by Shaw (1998), uses an elitist strategy to accept solutions (i.e., it accepts only improving solutions). On the other hand, the ALNS by Pisinger and Ropke (2007) uses the Metropolis criterion to accept solutions. According to this criterion, solutions are accepted with a given probability. If the newly found solution sol' improves the current solution sol the probability equals to one. Otherwise, the probability is computed using the Boltzmann expression: $e^{-(f(sol)-f(sol'))/\Upsilon}$. Parameter Υ is commonly known as the *temperature*. It is updated after each iteration using what is commonly known as the geometric cooling schedule: $\Upsilon = \Upsilon \times \bar{c}$, where $\bar{c} \in [0, 1)$. Then, the probability of accepting non-improving solutions decreases over the iterations. We tested the two approaches in our experiments.

We also tested a mix of them: We apply an elitist strategy during the first k iterations, and then, we activate the Metropolis criterion. We based our choice in two observations. First, using the elitist strategy, the search is often trapped in local optima after a certain amount of iterations, and then, it struggles to improve the solution. Second, as we do not ensure that our algorithm starts from a good quality solution, reaching a good solution can be time-consuming.

In our experiments, k is set to 300 and $\bar{c}$ to 0.9975. The initial temperature is fixed to $-\frac{0.25}{ln0.5} f(sol_0)$ where $f(sol_0)$ is the value of the objective function of the initial solution sol_0. Therefore, in the first iteration our approach accepts solutions that are 2.5% worse than the current solution with a probability of 0.5.

6 Computational experiments

6.1 Instances

Since our problem is new to the maintenance scheduling literature, no publicly available benchmarks exists. We therefore took advantage of our close collaboration with companies specializing on wind predictions, wind turbine maintenance and maintenance scheduling software, to get inside knowledge on how real data for the problem look like. Based on this knowledge, we built an instance generator that we believe captures reality with a good degree of accuracy.

We used our generator to build a 160-instance testbed (hereafter referred to simply as G1). For each instance, we consider time horizons of different lengths (10, 20 or 40), different number of time periods per day (2 or 4), different number of tasks (20, 40 or 80), and different number of skills (1 or 3). Each task can be executed in several modes (1 to 3). Note that $|\mathcal{S}| = 1$ simply means that no skills are considered. For each combination of parameters, we generate two categories of instances: 5 instances with a tight technician-to-work ratio (i.e., technicians can perform the majority of the tasks during the planning horizon, but they are not guar-

anteed to be enough to perform all the tasks) and 5 instances with a regular technician-to-work ratio (i.e., technicians can perform all the tasks during the planning horizon). We refer to the former as Type A and to the latter as Type B. We also refer to each family of instance with symbol "a_b_c_d_e" where a, b, c, d, and e refer to the number of time periods in the planning horizon, time periods within a day, skills, tasks, and to the technician-to-work ratio, respectively. For a thorough discussion on the instance generation process the reader is referred to "Appendix 3". Notice that in all our instances postponing a task is always non-profitable and therefore heavily penalized.

6.2 Results

We implemented our algorithms using *Java 8 (JVM 1.8.0.25)*. We rely on *Gurobi 6.5.1* for solving the ILP models $[P1]$ and $[P2]$ and *Choco 3.3.1* for solving the CP formulation (see Prud'homme et al. 2014). We ran our experiments on a Linux 64-bit machine, with an Intel(R) Xeon(R) X5675 (3.07 Ghz) and 12 GB of RAM.

Unless another formula is given (as for the results described in Tables 11, 13), all gaps reported in the article are computed as: $gap = (z^{UB} - z)/|z|$, where z is the objective function of the computed solution and z^{UB} is the objective function of the optimal solution or the minimal upper bound computed by Gurobi after 3 h of branch-and-bound when solving the two ILP formulations.

6.2.1 ILP formulations

First, we observe that the compact formulation $[P2]$ contains on average 1.5 times more variables (50 vs. 86 k) than the natural formulation $[P1]$, but 5 times less constraints (126 vs. 25 k). As shown below, this significantly impacts the performance of the solver. For reference, we point out that the set $\mathcal{P}$ contains 2400 plans on average.

Table 1 reports the average, over all the instances belonging to the same family, of: the gap (Gap), the solution time (Time), and the percentage of tasks scheduled (i.e., not posponed) in the best solution (%S). The table also reports the number of optimal solutions found within the 3-h time limit (#Opt). In order to have a meaningful comparison, the average solution time only takes into account those instances for which an optimal solution has been found within the time limit. Similarly, the average gap and percentage of scheduled tasks takes into account only the instances which are not optimally solved. This allows a better understanding of the results. Indeed, since in our instances postponing a task is heavily penalized, a large gap is often related to a low percentage of tasks scheduled during the time horizon. Notice that on average 99% of the tasks are scheduled in the optimal or best-known solutions for our testbed. To provide the reader with a different perspective, Table 2 presents the same results grouped by instance characteristic rather than by family of instances.

At a first glance, we observe that the compact formulation $[P2]$ outperforms the natural formulation $[P1]$ for small and medium-sized instances. For the large-sized instances, the two formulations struggle reaching optimal solutions, but the compact formulation performs worst than the natural one ($[P2]$ fails more often than $[P1]$ to schedule a large proportion of the tasks). We believe these results can be explained as follows. The compact formulation contains far less constraints than the natural formulation, and the value of the LP relaxation is around 2% smaller on average, which leads to tighter upper bounds computed by the ILP solver. We also observe that, at least in our 3-h time limit, optimality is only reached for small-sized instances and that whenever optimality is reached the CPU time is rather long (around 30 min on average). It is not very surprising as the formulations only involve binary variables and their size is quite large. We therefore reach the following conclusion: Solving the ILP formulations using a commercial solver does not yield suitable exact approaches for the problem.

Our results suggests that the number of skills does not have a significant impact on the difficulty of the instances (although we observe that instances with 3 skills appear to be easier to solve). This may be a result of less symmetries among technicians and a shorter number of feasible configurations to schedule the tasks. On the other hand, the number of tasks seems to have an impact on the difficulty of the instances when the technician-to-work ratio is tight. This can be explained by the higher difficulty of finding a maintenance plan when considering more tasks. It is also worth observing that the ILP formulations perform better on instances with 2 time periods per day; the solution time is shorter and the number of optimal solutions is larger than in those with 4 time periods per day. A plausible explanation is that the daily location-based incompatibilities are more binding on instances with more time periods per day. Indeed, a larger number of periods provides a wider choice of task starting times and therefore more opportunities to move technicians between locations during a single day. Instances with 4 time periods per day also have a larger number of plans and patterns; this may also explain their higher difficulty. In conclusion, according to our experiments, the difficulty of an instance increases with the number of time periods per day and the tightness of the technician-to-work ratio.

6.2.2 CP formulation

Table 3 summarizes the aggregated results found solving the CP model. In this experiment, we tested the resolution of the model with two branching strategies: regret-2 (R) and a randomized version of regret-2 coupled to a geometrical

Table 1 Computational results when solving the two ILP models (testbed G1—3-h time limit)

Family	[P1]				[P2]			
	Gap (%)	%S	#Opt	Time (s)	Gap (%)	%S	#Opt	Time (s)
10_2_1_20_A	1.4	97	0/5	–	1.8	95	4/5	1979
10_2_1_20_B	0.01	100	4/5	744	–	–	5/5	46
10_2_1_40_A	7.1	95	0/5	–	0.2	100	1/5	395
10_2_1_40_B	0.00	100	3/5	6015	–	–	5/5	343
10_2_3_20_A	2.0	96	1/5	326	0.9	100	4/5	1845
10_2_3_20_B	0.02	100	3/5	305	–	–	5/5	58
10_2_3_40_A	9.6	96	0/5	–	0.01	100	4/5	6477
10_2_3_40_B	0.00	100	4/5	2857	–	–	5/5	159
20_2_1_40_A	42	76	0/5	–	1.8	99	0/5	–
20_2_1_40_B	1.6	99	0/5	–	0.01	100	3/5	2888
20_2_1_80_A	28	87	0/5	–	436	0	0/5	–
20_2_1_80_B	6.2	96	0/5	–	67	67	2/5	4087
20_2_3_40_A	6.2	93	0/5	–	1.2	99	1/5	8149
20_2_3_40_B	0.02	100	2/5	2561	–	–	5/5	1213
20_2_3_80_A	23	89	0/5	–	48	79	0/5	–
20_2_3_80_B	4.3	97	0/5	–	196	50	3/5	2082
20_4_1_20_A	5.3	93	0/5	–	1.6	95	0/5	–
20_4_1_20_B	0.2	100	3/5	6264	–	–	5/5	666
20_4_1_40_A	161	46	0/5	–	264	0	0/5	–
20_4_1_40_B	12.8	91	0/5	–	131	50	1/5	1428
20_4_3_20_A	7.9	92	0/5	–	2.6	96	0/5	–
20_4_3_20_B	1.2	98	2/5	2152	–	–	5/5	1276
20_4_3_40_A	416	33	0/5	–	514	39	0/5	–
20_4_3_40_B	204	56	0/5	–	162	50	3/5	5877
40_4_1_40_A	147	54	0/5	–	309	18	0/5	–
40_4_1_40_B	157	73	0/5	–	430	40	0/5	–
40_4_1_80_A	49	80	0/5	–	4948	0	0/5	–
40_4_1_80_B	39	83	0/5	–	331	0	0/5	–
40_4_3_40_A	170	43	0/5	–	924	39	0/5	–
40_4_3_40_B	13	88	0/5	–	87	78	0/5	–
40_4_3_80_A	48	77	0/5	–	2813	0	0/5	–
40_4_3_80_B	24	84	0/5	–	3899	0	0/5	–

Table 2 Aggregated computational results when solving the two ILP models (testbed G1—3-h time limit)

Characteristic		[P1]				[P2]			
		Gap (%)	%S	#Opt	Time (s)	Gap (%)	%S	#Opt	Time (s)
$\lvert\mathcal{S}\rvert$	1	47	83	10/80	3981	636	44	26/80	1225
	3	68	81	12/80	1841	937	54	35/80	2252
$\frac{\lvert\mathcal{T}\rvert}{\lvert\mathcal{D}\rvert}$	2	10	94	17/80	2283	92	75	27/80	1743
	4	97	73	5/80	4619	1113	35	14/80	2055
Type	A	71	78	1/80	326	777	52	14/80	3553
	B	39	88	21/80	2932	763	42	47/80	1297
All		57	82	22/160	2814	773	48	61/160	1815

Table 3 Aggregated computational results when solving the CP model (testbed G1—average over 3 runs–5-min time limit)

Characteristic		R		R+restart	
		Gap (%)	%S	Gap (%)	%S
$\lvert\mathcal{S}\rvert$	1	9.2	94	2.7	98
	3	7.7	95	2.4	98
$\frac{\lvert\mathcal{T}\rvert}{\lvert\mathcal{D}\rvert}$	2	5.5	97	1.2	99
	4	11.4	93	4.0	97
Type	A	13.3	92	4.4	96
	B	3.6	98	0.8	100
All		8.4	95	2.6	98

restart policy (we restart the search from the root node) based on the number of backtracks (R+restart). The columns of the table report the relative average mean gap[5] (Gap) and the mean percentage of tasks scheduled in the solution[6] (*%S*) with 5 min of CPU time limit.

The results show that coupling our branching strategy with the restart policy gives the best results: The average gap is improved approximately by 6%. Jointly using a randomized branching strategy with a restart policy allows us to explore different parts of the search tree which increases the likelihood of finding better solutions. Table 4 reports additional results obtained solving the CP model with the R+restart configuration. We find good quality solutions for instances with 2 time periods per day and near-optimal solutions for Type B instances; the gap is larger for the other instances. We observe that the solutions obtained after a few iterations are little improved during the search. It seems that the CP model is facing some symmetry issues, especially on the technicians assignment. This drawback is not overcome with our restart policy. Nonetheless, we can notice that solving the CP formulation gives better overall results on the largest instances than solving the ILP formulations. Since the quality of the results are barely improved when increasing the time limit from 1 to 5 min, we did not consider necessary to test the model with a 3-h time limit as for the ILP formulations.

6.2.3 CPLNS

Our first experiment aimed to select the best acceptance criterion for our CPLNS. To achieve our goal, we ran our algorithm with three different solution acceptance criteria: elitism (El), Metropolis (MT), and both (El+MT) and three different time limits: 1, 3, and 5 min. Since the neighborhoods are partially randomized, we launched the algorithm 10 times for each instance. Table 5 shows that coupling an elitist strategy with the Metropolis acceptance criterion leads to the best results independently of the time limit. We there-

[5] Average of the mean gap found for each instance over 3 runs.

[6] Average of the mean percentage of tasks scheduled in the solution found for each instance over 3 runs.

Table 4 Detailed computational results when solving the CP model (testbed G1—R+restart configuration—average over 3 runs)

Family	1 min		3 min		5 min	
	Gap (%)	%S	Gap (%)	%S	Gap (%)	%S
10_2_1_20_A	1.3	98	1.3	98	1.3	98
10_2_1_20_B	0.4	100	0.4	100	0.4	100
10_2_1_40_A	2.4	99	2.4	99	2.4	99
10_2_1_40_B	0.4	100	0.4	100	0.4	100
10_2_3_20_A	2.0	97	2.0	97	2.0	97
10_2_3_20_B	0.3	100	0.3	100	0.3	100
10_2_3_40_A	2.4	99	2.3	99	1.9	99
10_2_3_40_B	0.8	100	0.8	100	0.8	100
20_2_1_40_A	2.5	99	2.5	99	2.5	99
20_2_1_40_B	0.3	100	0.3	100	0.3	100
20_2_1_80_A	3.2	99	3.2	99	3.2	99
20_2_1_80_B	0.2	100	0.2	100	0.2	100
20_2_3_40_A	1.6	99	1.6	99	1.6	99
20_2_3_40_B	0.2	100	0.2	100	0.2	100
20_2_3_80_A	1.2	100	1.2	100	1.1	100
20_2_3_80_B	0.3	100	0.2	100	0.2	100
20_4_1_20_A	2.4	95	2.2	95	2.1	95
20_4_1_20_B	1.0	100	0.9	100	0.9	100
20_4_1_40_A	9.9	93	9.6	93	9.5	93
20_4_1_40_B	2.9	99	2.9	99	2.9	99
20_4_3_20_A	6.3	94	5.7	94	5.7	94
20_4_3_20_B	1.7	99	1.7	99	1.7	99
20_4_3_40_A	6.9	94	6.9	94	6.9	94
20_4_3_40_B	1.9	99	1.8	99	1.8	99
40_4_1_40_A	7.8	94	7.8	94	7.8	94
40_4_1_40_B	0.8	100	0.7	100	0.5	100
40_4_1_80_A	9.7	94	8.8	95	8.8	95
40_4_1_80_B	0.5	100	0.5	100	0.5	100
40_4_3_40_A	5.5	96	5.5	96	5.4	96
40_4_3_40_B	0.6	100	0.6	100	0.6	100
40_4_3_80_A	8.0	95	7.9	95	7.9	95
40_4_3_80_B	0.5	100	0.4	100	0.4	100
All	2.7	98	2.6	98	2.6	98

Table 5 Average computational results according to the solution acceptance criterion (testbed G1—average over 10 runs)

Time limit	Average gap		
	1 min (%)	3 min (%)	5 min (%)
El	1.54	1.33	1.26
MT	1.58	1.34	1.25
El + MT	1.54	1.30	1.21

Table 6 Computational results for the CPLNS (testbed G1—average over 10 runs)

Family	1 min		3 min		5 min	
	Gap (%)	%S	Gap (%)	%S	Gap (%)	%S
10_2_1_20_A	0.8	98	0.8	98	0.8	98
10_2_1_20_B	0.0	100	0.0	100	0.0	100
10_2_1_40_A	1.0	100	0.5	100	0.4	100
10_2_1_40_B	0.0	100	0.0	100	0.0	100
10_2_3_20_A	0.8	99	0.7	99	0.6	99
10_2_3_20_B	0.0	100	0.0	100	0.0	100
10_2_3_40_A	1.5	99	1.1	99	1.0	99
10_2_3_40_B	0.1	100	0.1	100	0.0	100
20_2_1_40_A	1.4	99	1.0	99	0.9	99
20_2_1_40_B	0.1	100	0.0	100	0.0	100
20_2_1_80_A	3.1	98	2.5	99	2.2	99
20_2_1_80_B	0.1	100	0.1	100	0.1	100
20_2_3_40_A	0.9	99	0.6	100	0.6	100
20_2_3_40_B	0.0	100	0.0	100	0.0	100
20_2_3_80_A	0.5	100	0.4	100	0.3	100
20_2_3_80_B	0.1	100	0.1	100	0.1	100
20_4_1_20_A	1.1	95	1.1	95	1.0	95
20_4_1_20_B	0.1	100	0.1	100	0.1	100
20_4_1_40_A	6.2	94	5.4	94	5.3	94
20_4_1_40_B	1.2	99	0.5	100	0.2	100
20_4_3_20_A	3.1	95	2.6	96	2.0	96
20_4_3_20_B	0.3	100	0.2	100	0.2	100
20_4_3_40_A	4.7	95	4.0	95	3.9	95
20_4_3_40_B	0.8	100	0.6	100	0.5	100
40_4_1_40_A	4.7	96	4.5	96	4.4	96
40_4_1_40_B	0.2	100	0.2	100	0.1	100
40_4_1_80_A	5.8	96	5.4	96	5.4	96
40_4_1_80_B	0.3	100	0.2	100	0.2	100
40_4_3_40_A	3.8	97	3.1	98	2.8	98
40_4_3_40_B	0.2	100	0.2	100	0.1	100
40_4_3_80_A	6.3	95	5.9	95	5.6	96
40_4_3_80_B	0.2	100	0.2	100	0.2	100

fore used this acceptance criterion in the remainder of our experiments.

We now discuss more thoroughly the performance of the CPLNS algorithm. Table 6 reports the results delivered by our CPLNS for each family of instances. The columns in the table report the relative average mean gap[7] (Gap) and the mean percentage of tasks scheduled in the solution[8] (%S) running with a 1, 3, and 5 min of CPU time limit. These experiments aim to enable a decision maker to define a CPU time limit according to the trade-off between resolution time and quality of the results he or she is interested in. To provide the reader with a different perspective, Table 7 presents the same results grouped by instance characteristic rather than by family of instances.

Table 7 Aggregated computational results for the CPLNS (testbed G1—average over 10 runs)

Characteristic		1 min		3 min		5 min	
		Gap (%)	%S	Gap (%)	%S	Gap (%)	%S
$\lvert\mathcal{S}\rvert$	1	1.6	98	1.4	99	1.3	99
	3	1.5	99	1.2	99	1.1	99
$\frac{\lvert\mathcal{T}\rvert}{\lvert\mathcal{D}\rvert}$	2	0.7	99	0.5	100	0.4	100
	4	2.4	98	2.1	98	2.0	98
Type	A	2.9	97	2.5	97	2.3	97
	B	0.2	100	0.2	100	0.1	100
All		1.5	99	1.3	99	1.2	99

Since the gap is computed with respect to upper bounds for the largest instances, assessing the intrinsic quality of the CPLNS using only the gap is sometimes not conclusive enough. However, the overall average gaps of 1.2% after 5 min show the effectiveness of our approach. We might expect to be closer to the optimal solutions for the largest instances. The algorithm provides near-optimal solutions for all the Type B instances and all the instances where the number of time periods per day is equal to 2, but the performance is slightly inferior on Type A instances in which the number of time periods per day is equal to 4. This last observation can be explained by the fact that the number of plans and thus the model to be considered by the CP model when repairing the solution is larger. Since we only allow a limited number of backtracks, the quality of the first decisions taken in our branching strategies strongly impacts the capacity of the algorithm to improve the current solutions. The algorithm may then sometimes fail building better solutions with the CP model (in the reparation stage), although it could be possible if it explored the whole search space.

Table 8 reports the relative average mean gap (Mean), the average best gap[9] (Best) and the average worst gap[10] (Worst) for the CPLNS with 5 min of CPU time limit (detailed results are available in Table 10 in "Appendix 2"). The CPLNS exhibits a stable behavior: on average the difference between the best and the worst solution found over the 10 runs is a reduced 0.68%.

[7] Average of the mean gap found for each instance over 10 runs.

[8] Average of the mean percentage of tasks scheduled in the solution found for each instance over 10 runs.

[9] Average of the best (minimal) gap found for each instance over 10 runs.

[10] Average of the worst (maximal) gap found for each instance over 10 runs.

Table 8 Behavior of the CPLNS on the 10 runs (testbed G1—5 min time limit)

Characteristic		Mean (%)	Best (%)	Worst (%)
$\lvert\mathcal{S}\rvert$	1	1.32	1.04	1.65
	3	1.12	0.76	1.50
$\frac{\lvert\mathcal{T}\rvert}{\lvert\mathcal{D}\rvert}$	2	0.45	0.25	0.66
	4	2.00	1.55	2.50
Type	A	2.33	1.75	2.93
	B	0.12	0.05	0.22
All		1.22	0.90	1.58

For each approach presented in the article to solve the problem, we report in Table 11 of "Appendix 2" the gap to the best solution (lower bound) that we were able to compute in all the tests presented in this section. This gap is computed as: $gap = (z^{LB} - z)/|z|$, where z is the objective function of the computed solution and z^{LB} is the best solution (i.e., best lower bound) found throughout our tests. For small-sized instances, the direct resolution of the ILP formulations gives the best solutions. Nonetheless, for those instances, the solutions obtained from the resolution of the CP model and from the CPLNS are very close. For medium-sized and large-sized instances, the CPLNS outperforms the other approaches.

7 Particular case: handling of corrective tasks

Up to this point, we have only considered preventive tasks. Sometimes, however, the technicians may also have to perform corrective tasks. Our models and methods can be easily adapted to deal with corrective tasks in a number of practical situations. We describe in this section those adaptations. Needless to say, since our approaches are conceived to work on a static case, we only focus on how to handle non-started corrective tasks that are known with certitude prior to the beginning of the planning horizon.

7.1 Extending models and methods

First, let us introduce new binary parameters. We denote $\ddot{b}_{wi} = 1$ if and only if task i is a corrective task that shuts down turbine w until it is not entirely completed. Our models require the following minor modifications.

We add constraints (55) and (56) to ILP formulation [$P1$].

$$f_w^t + 1 - \sum_{t' \in \mathcal{T} \text{ s.t. } t' \leq t} c_i^{t'} \leq 1 \quad \forall w \in \mathcal{W},$$
$$\forall i \in \mathcal{I} \text{ s.t. } \ddot{b}_{wi} = 1, \quad \forall t \in \mathcal{T}, \tag{55}$$
$$\widetilde{f}_w^d + 1 - \sum_{t' \in \mathcal{T} \text{ s.t. } t' \leq t_d^{rest}+1} c_i^{t'} \leq 2 \quad \forall w \in \mathcal{W},$$
$$\forall i \in \mathcal{I} \text{ s.t. } \ddot{b}_{wi} = 1, \quad \forall d \in \mathcal{D}, \tag{56}$$

Constraints (55) and (56) state that a turbine w is unavailable during a time period (or a rest time period) if there are incomplete corrective tasks related to that turbine.

In a similar vein, we add constraints (57) and (58) to ILP formulation [$P2$].

$$f_w^t + \sum_{p \in \mathcal{P}_i \text{ s.t. } \max_{t' \in \mathcal{T}} a_p^{t'} < t} \ddot{b}_{wi} x_p \leq 1$$
$$\forall w \in \mathcal{W}, \quad \forall i \in \mathcal{I}, \quad \forall t \in \mathcal{T}, \tag{57}$$
$$\widetilde{f}_w^d + \sum_{p \in \mathcal{P}_i \text{ s.t. } \max_{t' \in \mathcal{T}} a_p^{t'} \leq t_d^{rest}} \ddot{b}_{wi} x_p \leq 1$$
$$\forall w \in \mathcal{W},$$
$$\forall i \in \mathcal{I} \text{ s.t. } \ddot{b}_{wi} = 1, \quad \forall d \in \mathcal{D}, \tag{58}$$

In the CP model, we replace constraints (48) and (50) by the following constraints:

$$t \leq C_i \Rightarrow t \notin F_w^{day}$$
$$\forall w \in \mathcal{W}, \quad \forall i \in \mathcal{I} \text{ s.t. } \ddot{b}_{wi} = 1, \quad \forall t \in \mathcal{T}, \tag{59}$$
$$\left(t \notin \bigcup_{i \in \mathcal{I} | b_{wi} = 1} E_i\right) \wedge \left(\bigwedge_{i \in \mathcal{I} | \ddot{b}_{wi} = 1} (t > C_i)\right) \Rightarrow t \in F_w^{day}$$
$$\forall w \in \mathcal{W}, \quad \forall t \in \mathcal{T}, \tag{60}$$
$$(t_d^{rest} + 1) \leq C_i \Rightarrow d \notin F_w^{rest} \quad \forall w \in \mathcal{W},$$
$$\forall i \in \mathcal{I}, \text{ s.t. } \ddot{b}_{wi} = 1, \quad \forall d \in \mathcal{D}, \tag{61}$$
$$\left(\bigwedge_{i \in \mathcal{I} | \widetilde{b}_{wi} = 1} \left(\{t_d^{rest}, t_d^{rest} + 1\} \nsubseteq E_i\right)\right)$$
$$\wedge \left(\bigwedge_{i \in \mathcal{I} | \ddot{b}_{wi} = 1} \left(t_d^{rest} + 1 > C_i\right)\right) \Rightarrow d \in F_w^{rest}$$
$$\forall w \in \mathcal{W}, \quad \forall d \in \mathcal{D} \tag{62}$$

Constraints (59) and (61) state that a turbine w is unavailable during a time period if there are incomplete corrective tasks related to that turbine. Constraints (60) and (62) ensure that a turbine is available to produce electricity during a time period if and only if i) no preventive tasks requiring its shutdown are scheduled during the time period and ii) if all the corrective tasks are completed. Note that by adapting the CP model to work with corrective tasks, the CPLNS is also automatically adapted to this new scenario.

7.2 Practical applications

Our adapted models and CPLNS can be used to deal with different practical situations. For illustration purposes, in the remainder of this subsection we briefly discuss two of them.

Table 9 Aggregated computational results for the four different approaches (testbed G2)

Characteristic		[P1]				[P2]				CP		CPLNS	
		3h Gap (%)	%S	#Opt	Time (s)	3h Gap (%)	%S	#Opt	Time (s)	5min Gap (%)	%S	5min Gap (%)	%S
$\|\mathcal{S}\|$	1	150	81	9/80	3976	200	62	31/80	1516	7.0	98	4.1	98
	3	83	80	15/80	3351	92	58	38/80	1665	7.2	99	2.9	99
$\frac{\|\mathcal{T}\|}{\|\mathcal{D}\|}$	2	17	94	21/80	3757	29	88	52/80	1431	1.5	99	0.5	100
	4	196	70	3/80	2382	205	48	17/80	2109	12.7	97	6.5	98
Type	A	174	72	5/80	5572	181	59	17/80	1458	12.7	97	6.9	98
	B	50	90	19/80	3062	82	64	52/80	1644	1.5	100	0.1	100
All		118	81	24/160	3585	150	60	69/160	1598	7.1	98	3.5	99

S1—corrective first, preventive second: As mentioned earlier, in the wind energy industry, more often than not, the maintenance is performed by contractors rather than by the wind farm owners. Therefore, it is difficult for maintenance companies to delay the execution of corrective tasks without generating a conflict with their customers (especially if the breakdown impedes energy production). In these situations, corrective tasks are simply scheduled first and, usually, as soon as possible. Our approaches can easily deal with this situation. The decision maker only needs to solve two independent problems (one for the corrective tasks followed by one for the preventive tasks), adjusting on the second the parameters π_r^t and l_r^t according to the pre-established crew assignments. We can also decide to reconsider these assignments when scheduling the preventive tasks. To this end, for the second stage of the optimization, we add to set $\mathcal{I}$ the corrective tasks along with the preventive tasks, but we enforce the former to be scheduled as found during the first stage of the optimization.

S2—corrective: + preventive: In some cases, scheduling corrective tasks as early as possible tasks may lead to bad overall decisions. Indeed, if the wind speed is too low to produce electricity for many consecutive days, there is some flexibility to schedule these tasks latter and use the resources (i.e., technicians) to perform tasks that would be optimally scheduled at the beginning of the planning horizon. For instance, assume that a given corrective task has a duration of 3 time periods and induces a very small loss of revenue (or none at all) if we wait, say, two time periods to perform it (the forecasted wind is very low from the beginning of the time horizon until the sixth time period). Assume also that there is a preventive maintenance task to perform at another location where the revenue is very low for the two first time periods but very large after this. If the two tasks cannot be executed in parallel (e.g., lack of technicians), it is probably more profitable to schedule first the preventive task and then the corrective task. In this case, the decision maker may want to combine corrective and preventive tasks and solve a unique problem. This case can be tackled simply by adding both preventive and corrective tasks to set $\mathcal{I}$ and running our models and/or CPLNS with the modifications introduced above.

7.3 Some results

To assess the impact of considering corrective tasks in the performance (quality and speed) of our approaches, we conducted experiments on a new set of instances mixing preventive and corrective tasks. It is worth recalling that the objective of our research is not to compare managerial practices in maintenance scheduling; therefore, we limited our experiments in this section to practical situation S2. We believe this situation is the most general and difficult to solve from a purely computational performance perspective. Moreover, to a certain extent, we have already handled the practical situation S1 in the experiments described in the previous section. Our new testbed, denoted G2, is composed from the same families than those in G1. In the instance generation process, the probability of generating a corrective task was set equal to 5%. Nonetheless, we ensured that each instance in this testbed contains at least one corrective task. If a task is corrective, it can prevent more than one turbine to produce electricity with a probability of 7.5%. Similarly than for testbed G1, the penalty of postponement is set in such a way that postponing a task is never profitable. Notice that the penalty is larger than for testbed G1 as the maximal loss of revenue than can be induced by the scheduling of a corrective task is larger.

In general the results obtained in G2 confirm our findings. Solving directly the ILP formulations is just suitable for small-sized instances, while solving the CP formulation gives solutions with a reduced gap. However, the CPLNS remains the best overall approach. Table 9 summarizes the results. Detailed results for each family of instances can be found in Appendix "Testbed G2" section.

8 Conclusions and research perspectives

In this study, we introduced a new and challenging maintenance scheduling problem faced by the wind power industry. Some of the special features of this problem are the existence of alternative execution modes for each task and the individual management of the technicians through a space–time tracking. We also introduced an original objective function, far from the classical scheduling concerns, linking the revenue to the periods during which the maintenance operations are performed.

We proposed three mathematical formulations based on both constraint and integer linear programming. Computational results indicate that, generally, the models cannot be directly used to solve realistic instances. ILP models are unable to solve to optimality most of the instances after 3 h and the gap is still very large for many families of instances. Nevertheless, we showed that an ILP compact formulation using the notion of plans outperforms the more natural ILP formulation of the problem. Moreover, results indicate that the CP model produces high quality solutions for small-sized instances. However, it does not yield very good results, in general, for the majority of medium and large-sized instances. The performance of our CP model actually seems to be affected by symmetry issues, especially on the technicians assignment.

To provide an alternative solution approach, we developed a CP-based large neighborhood search. We successfully adapted some destroy operators to this new problem and proposed some new ones. Moreover, we designed several branching strategies to effectively repair solutions solving a CP model with fixed variables. The CPLNS shows an average gap of 1.2% with respect to the optimal solutions if known, or to the best-known upper bounds otherwise. It provides near-optimal solutions when the technician-to-work ratio is regular, whereas, when it is tight, the gap increases as the problem size (number of tasks and number of time periods per day) grows. Nonetheless, the computational results demonstrate the efficiency of the proposed method. We have also showed how our method can also handle corrective tasks known prior to the beginning of the planning horizon.

As a perspective, we could extend the definition of the problem to handle the case of technicians working different shifts. Two different approaches could be considered: either one allows tasks to be initiated by some technicians and finished by other technicians (since tasks can last more than the duration of a shift) or one restricts tasks to be performed by technicians working the same shift. To our knowledge, the former is not common in practice, so the latter may be more relevant. The second proposition is compatible—with slight adjustments—with the compact formulation [$P2$]. Indeed, every plan would be associated with a single shift (we would not create any plan that overlaps two different shifts) and we would restrict technicians to be assigned to plans corresponding to their shift. The CPLNS would require a broader change as the duration of a task would depend on the number of days it overlaps (the duration would become dependent on the starting time in addition to the execution mode). Moreover, we could consider the case of tasks requiring technicians with complementary skills.

Future works also include the development of efficient exact approaches. One can observe an intrinsic decomposition of the problem into a scheduling problem on the one hand and a resource management problem on the other hand. This leads us to investigate a branch-and-check approach as well as cut generation processes. Last but not least, we have only addressed the deterministic problem, but, as a matter of fact, the wind speed (and therefore the revenue) is stochastic by nature.

Acknowledgements The authors would like to kindly thank two anonymous reviewers for their comments and suggestions. This work was supported by Angers Loire Métropole through its research grant program; and by the Natural Sciences and Engineering Research Council of Canada (NSERC) through a grant that enabled the collaboration with the Canadian company WPred, which we would like to thank for their expertise.

Appendix 1: Notations

Time

$\mathcal{T}$: time horizon
$\mathcal{D}$: set of days
$\mathcal{T}_d$: set of time periods that belong to day $d \in \mathcal{D}$
t_d^{rest}: last time period $t \in \mathcal{T}$ before the rest time period following day $d \in \mathcal{D}$

Locations

$\mathcal{L}$: set of locations (wind farms and technician home depots)
$\delta_{ll'}$: binary parameter equal to 1 if and only if locations l and l' are compatible

Technicians

$\mathcal{S}$: set of skills
$\mathcal{R}$: set of technicians

λ_{rs}: binary parameter equal to 1 if and only if technician $r \in \mathcal{R}$ masters skill $s \in \mathcal{S}$
π_r^t: binary parameter equal to 1 if and only if technician $r \in \mathcal{R}$ is available during time period $t \in \mathcal{T}$
l_r^t: location of technician $r \in \mathcal{R}$ when he or she is not available during time period $t \in \{t' \in \mathcal{T}, \pi_r^{t'} = 1\}$

Tasks

$\mathcal{I}$: set of tasks to be performed on the turbines
$ov(\mathcal{I})$: set of subsets of non-overlapping tasks

l_i: location where task $i \in \mathcal{I}$ has to be performed
$\mathcal{M}_i$: set of execution modes for task $i \in \mathcal{I}$
m_i^0: execution mode related to the postponement of task $i \in \mathcal{I}$ ($m_i^0 \in \mathcal{M}_i$)
q_{im}: number of technicians required during each time period to perform task $i \in \mathcal{I}$ in mode $m \in \mathcal{M}_i$ ($q_{im_i^0} = 0$)
d_{im}: duration of task $i \in \mathcal{I}$ if performed in mode $m \in \mathcal{M}_i$ ($d_{im_i^0} = 0$)

γ_i^t: binary parameter equal to 1 if and only if task $i \in \mathcal{I}$ can be executed during time period $t \in \mathcal{T}$
s_i: skill required to perform task $i \in \mathcal{I}$
o_i: penalty if task $i \in \mathcal{I}$ is postponed.

Turbines

$\mathcal{W}$: set of turbines

b_{wi}: binary parameter equal to 1 if and only if the execution of task $i \in \mathcal{I}$ shuts down turbine $w \in \mathcal{W}$ when technicians are effectively working on i
$\widetilde{b}_{wi}$: binary parameter equal to 1 if and only if the execution of task $i \in \mathcal{I}$ shuts down turbine $w \in \mathcal{W}$ during the rest time periods it overlaps
g_w^t: revenue if turbine $w \in \mathcal{W}$ can produce electricity during time period $t \in \mathcal{T}$
$\widetilde{g}_w^d$: revenue if turbine $w \in \mathcal{W}$ can produce electricity during the rest time period following day $d \in D$

Plans

$\mathcal{P}$: set of plans
$\mathcal{P}_i$: set of plans involving task $i \in \mathcal{I}$
i_p: task involved in plan $p \in \mathcal{P}$
a_p^t: binary parameter equal to 1 if and only if task i_p is executed during time period $t \in \mathcal{T}$
q_p: number of required technicians if plan $p \in \mathcal{P}$ is selected
$\mathcal{R}_p = \mathcal{R}_{i_p}$
$b_{wp} = b_{wi_p}$
$\widetilde{b}_{wp} = \widetilde{b}_{wi_p}$
$o_p = o_{i_p}$

Appendix 2: Detailed computational results

Testbed G1

See Tables 10 and 11.

Testbed G2

Table 12 reports the results obtained when solving the ILP and CP models and when running the CPLNS. To assess the quality of the results, it is noteworthy that large values for the gap are essentially associated with Type A instances. Because the technician-to-work ratio is tighter, it is more difficult to schedule all the tasks. Scheduling one additional task would drastically reduce this gap.

From the results, we do not see any significant impact of adding corrective tasks on the difficulty to solve the instances. It is not so surprising as corrective tasks usually need to be scheduled at the beginning of the planning horizon in order not to induce too much lost revenue. For our models, this might implies less symmetry on the scheduling of the tasks. Moreover, we can draw identical conclusions on the efficiency of the approaches. Solving the ILP formulations does not lead to an efficient exact approach as only the small-sized instances can be optimally solved, whereas solving the CP formulation yields good results for Type B instances. The CPLNS gives the overall best results and near-optimal

Table 10 Behavior of the CPLNS on the 10 runs (testbed G1—5 min time limit)

Family	Mean (%)	Best (%)	Worst (%)
10_2_1_20_A	0.75	0.70	0.87
10_2_1_20_B	0.03	0.00	0.03
10_2_1_40_A	0.42	0.11	0.95
10_2_1_40_B	0.02	0.01	0.03
10_2_3_20_A	0.60	0.37	0.79
10_2_3_20_B	0.01	0.01	0.03
10_2_3_40_A	1.05	0.22	1.78
10_2_3_40_B	0.05	0.01	0.08
20_2_1_40_A	0.86	0.20	1.68
20_2_1_40_B	0.04	0.02	0.09
20_2_1_80_A	2.24	1.49	2.92
20_2_1_80_B	0.06	0.04	0.08
20_2_3_40_A	0.58	0.47	0.70
20_2_3_40_B	0.02	0.01	0.04
20_2_3_80_A	0.34	0.26	0.43
20_2_3_80_B	0.07	0.05	0.09
20_4_1_20_A	1.03	0.96	1.28
20_4_1_20_B	0.09	0.01	0.21
20_4_1_40_A	5.30	4.52	5.81
20_4_1_40_B	0.24	0.09	0.87
20_4_3_20_A	2.02	1.03	3.52
20_4_3_20_B	0.23	0.04	0.42
20_4_3_40_A	3.86	2.78	4.35
20_4_3_40_B	0.46	0.17	0.71
40_4_1_40_A	4.37	4.26	4.50
40_4_1_40_B	0.13	0.07	0.20
40_4_1_80_A	5.38	3.99	6.67
40_4_1_80_B	0.23	0.16	0.30
40_4_3_40_A	2.79	1.71	3.95
40_4_3_40_B	0.12	0.06	0.19
40_4_3_80_A	5.62	4.86	6.76
40_4_3_80_B	0.17	0.12	0.22
All	1.22	0.90	1.58

Table 11 Gap to the best lower bound according to the different approaches (testbed G1)

Family	[P1]	[P2]	CP	CPLNS
	3 h (%)	3 h (%)	5 min (%)	5 min (%)
10_2_1_20_A	1.1	**0.0**	0.9	0.4
10_2_1_20_B	**0.0**	**0.0**	0.4	**0.0**
10_2_1_40_A	7.0	0.1	2.4	0.4
10_2_1_40_B	**0.0**	**0.0**	0.4	**0.0**
10_2_3_20_A	1.4	**0.0**	1.8	0.4
10_2_3_20_B	**0.0**	**0.0**	0.3	**0.0**
10_2_3_40_A	9.6	**0.0**	1.8	1.0
10_2_3_40_B	**0.0**	**0.0**	0.8	**0.0**
20_2_1_40_A	40.9	1.0	1.6	**0.1**
20_2_1_40_B	1.6	**0.0**	0.3	0.0
20_2_1_80_A	24.9	432.2	1.0	**0.0**
20_2_1_80_B	6.2	40.0	0.2	**0.0**
20_2_3_40_A	5.7	0.4	1.1	**0.1**
20_2_3_40_B	**0.0**	**0.0**	0.2	**0.0**
20_2_3_80_A	22.4	47.5	0.7	**0.0**
20_2_3_80_B	4.2	78.4	0.2	**0.0**
20_4_1_20_A	4.3	0.5	1.1	**0.0**
20_4_1_20_B	0.1	**0.0**	0.9	0.1
20_4_1_40_A	152.2	256.4	4.0	**0.0**
20_4_1_40_B	12.6	104.7	2.6	**0.0**
20_4_3_20_A	6.2	0.9	4.0	**0.4**
20_4_3_20_B	0.7	**0.0**	1.7	0.2
20_4_3_40_A	403.6	497.5	3.1	**0.2**
20_4_3_40_B	204.1	64.7	1.8	**0.4**
40_4_1_40_A	138.0	295.7	3.3	**0.0**
40_4_1_40_B	157.0	429.8	0.4	**0.0**
40_4_1_80_A	40.6	4777.1	3.2	**0.0**
40_4_1_80_B	38.7	330.1	0.3	**0.0**
40_4_3_40_A	165.1	897.5	2.5	**0.0**
40_4_3_40_B	13.4	86.9	0.5	**0.0**
40_4_3_80_A	39.3	2731.3	2.1	**0.0**
40_4_3_80_B	23.8	3893.2	0.3	**0.0**
All	47.7	467.7	1.4	**0.1**

Bold values indicate the best approach for the given family of instances

solutions for small and medium-sized instances. Lastly, the difficulty of the instances seems also to be related to the same characteristic as for testbed G1.

For each approach, we also report in Table 13 the gap to the best solution that we were able to compute in our tests. We find similar results as for testbed G1. Globally, the CPLNS outperforms the other approaches.

Appendix 3: Instance generation

An instance of the problem is primarily characterized by:

- a finite time horizon (a finite number of time periods)
- a number of time periods per day (yielding the number of days)
- a set of locations (wind farms + home depot)
- a set of wind turbines distributed over the wind farms
- a set of maintenance tasks to perform at the different locations and that impact the availability of the turbines
- a set of technicians to perform the tasks
- wind speed for each time period and location
- postponing penalties

The generator is based on the following parameters:

- $n_{\mathcal{T}}, n_{\mathcal{D}}, n_{\mathcal{I}}, n_{\mathcal{S}}$ (length of time horizon, number of days, number of wind farms, number of tasks and number of skills)
- $Dn^{\mathcal{L}}$: probability distribution of the number of locations
- Dl^{xy}: probability distribution of the coordinates associated with each location
- $Dn^{\mathcal{L}}_{\mathcal{W}}$: probability distribution of the number of turbines per location
- $Dn^{\mathcal{W}}_{\mathcal{I}}$: probability distribution of the number of tasks per turbine
- Δl^{min}: minimum distance between two locations
- Δr^{max}: maximum distance between two locations such that they can be visited by the a technician during the same day
- K: set of all types of preventive tasks that we consider
- $p(k)$: probability of generating a task of type $k \in K$
- $Di_{impact}(k)$: probability distribution of the impact of each type of preventive task on the wind turbines
- $Di_{dur}(k)$: probability distribution of the duration of each type of preventive task
- $Di_{req}(k)$: probability distribution of the number of technicians that can perform each type of preventive task during any time period
- $Dr_{\#skills}$: probability distribution of the number of skills mastered by a technician
- $Dr_{\mathbb{P}(unv)}$: probability that a technician has some unavailability time periods during the time horizon
- $Dr_{\#unv}$: probability distribution of the number of time periods during which a technician is unavailable
- $Dr_{d^{unv}}$: probability distribution of the duration of the unavailability of a technician (in man-hours)
- Dw_{power}: probability distribution of the nominal power (in kW) of each turbine
- $\widehat{\phi}$: average wind speed on each wind farm
- Υ^{safety}_{max}: maximum wind speed allowed to perform a task
- Δl^{max}: maximum distances for the spatial correlation of the wind speed
- δ: number of values used in the moving average for the timewise dependency between the wind speeds
- α: correlation factor between wind speed

We generate an instance following multiple steps. First of all, the length of time horizon, the number of days, the number

Table 12 Computational results for the four different approaches (testbed G2)

Family	[P1]				[P2]				CP		CPLNS	
	3 h				3 h				5 min		5 min	
	Gap (%)	%S	#Opt	Time (s)	Gap (%)	%S	#Opt	Time (s)	Gap (%)	%S	Gap (%)	%S
10_2_1_20_A	2.5	98	1/5	9877	4.9	100	4/5	991	3.0	98	1.8	99
10_2_1_20_B	0.01	100	4/5	2082	–	–	5/5	110	1.2	100	0.01	100
10_2_1_40_A	17	95	0/5		0.2	100	1/5	315	2.9	99	1.2	99
10_2_1_40_B	2.6	98	2/5	6083	–	–	5/5	970	1.8	99	0.04	100
10_2_3_20_A	0.01	100	4/5	4496	–	–	5/5	82	0.7	100	0.2	100
10_2_3_20_B	–	–	5/5	3110	–	–	5/5	25	1.2	100	0.02	100
10_2_3_40_A	48	79	0/5		0.01	100	4/5	4767	4.0	99	0.7	100
10_2_3_40_B	0.1	100	1/5	738	–	–	5/5	2259	0.7	100	0.1	100
20_2_1_40_A	20	91	0/5		2.5	99	0/5	–	3.4	98	1.8	99
20_2_1_40_B	0.1	100	1/5	2929	–	–	5/5	987	0.1	100	0.01	100
20_2_1_80_A	62	89	0/5		70	78	0/5	–	2.5	99	1.5	99
20_2_1_80_B	12	97	0/5		0.1	100	2/5	1598	0.4	100	0.1	100
20_2_3_40_A	6.8	98	0/5		1.2	99	1/5	353	0.6	100	0.1	100
20_2_3_40_B	0.04	100	3/5	3776	–	–	5/5	732	0.8	100	0.03	100
20_2_3_80_A	24	89	0/5		87	59	0/5	–	1.4	99	0.8	100
20_2_3_80_B	7.4	97	0/5		–	–	5/5	4337	0.1	100	0.03	100
20_4_1_20_A	5.0	94	0/5		1.6	96	0/5	–	3.9	95	2.1	95
20_4_1_20_B	0.5	99	0/5		–	–	5/5	3166	2.9	98	0.1	99
20_4_1_40_A	185	18	0/5		74	56	0/5	–	14	94	7.0	95
20_4_1_40_B	8.4	96	1/5	2485	0.30	100	3/5	3680	1.8	99	0.1	100
20_4_3_20_A	3.3	96	0/5		3.5	95	2/5	334	6.9	96	2.2	97
20_4_3_20_B	2.2	100	2/5	2331	–	–	5/5	183	3.5	100	0.7	99
20_4_3_40_A	444	48	0/5		94	20	0/5	–	69	95	29	96
20_4_3_40_B	47	75	0/5		0.7	99	1/5	5078	6.4	99	0.2	100
40_4_1_40_A	1308	30	0/5		1368	0	0/5	–	15	95	9.2	95
40_4_1_40_B	66	73	0/5		0.4	100	1/5	2325	1.0	100	0.1	100
40_4_1_80_A	91	73	0/5		268	0	0/5	–	58	94	41	95
40_4_1_80_B	355	84	0/5		176	0	0/5	–	0.5	100	0.2	100
40_4_3_40_A	249	16	0/5		53	77	0/5	–	6.3	97	2.9	98
40_4_3_40_B	55	77	0/5		1.5	99	0/5	–	1.0	100	0.1	100
40_4_3_80_A	139	76	0/5		256	0	0/5	–	11	95	9.1	96
40_4_3_80_B	60	88	0/5		283	0	0/5	–	0.9	100	0.1	100
All	118	81	24/160	3585	150	60	69/160	1598	7.1	98	3.5	99

of tasks and the number of skills are input values. This yields directly the set $\mathcal{T}$ of time periods and the set $\mathcal{D}$ of days.

We then start the generation of an instance by building the set $\mathcal{L}$ of locations whose cardinality is set by sampling the $Dn^{\mathcal{L}}$ distribution. According to the distance Δl^{min}, we then generate the coordinates of each location by sampling the Dl^{xy} distribution. Based on these coordinates and on the distance Δr^{max}, we compute the parameters $(\delta_{ll'})_{(l,l')\in\mathcal{L}^2}$ that enable to define the daily location-based incompatibility constraints.

Afterward, we built the set $\mathcal{W}$ of wind turbines. To this end, according to the target number of tasks, we start by generating a number of wind turbines per locations by sampling the $Dn^{\mathcal{L}}_{\mathcal{W}}$ distribution. For each location where there is at least one wind turbine (i.e., this location is a wind farm), we then generate a nominal power by sampling the Dw_{power} distribution and we set the nominal power P_w equal to this latter value for each wind turbine $w \in \mathcal{W}$ of the wind farm.

After that, we call procedure **genTasks**() to create the set $\mathcal{I}$ of tasks. Notice that for each task $i \in \mathcal{I}$ we build the set

Table 13 Gap to the best lower bound according to the different approaches (testbed G2)

Family	[P1]	[P2]	CP	CPLNS
	3 h (%)	3 h (%)	5 min (%)	5 min (%)
10_2_1_20_A	1.0	**0.0**	2.0	0.8
10_2_1_20_B	**0.0**	**0.0**	1.1	**0.0**
10_2_1_40_A	16.8	**0.0**	2.7	1.0
10_2_1_40_B	1.6	**0.0**	1.8	**0.0**
10_2_3_20_A	**0.0**	**0.0**	0.7	0.2
10_2_3_20_B	**0.0**	**0.0**	1.2	**0.0**
10_2_3_40_A	47.7	**0.0**	4.0	0.7
10_2_3_40_B	0.1	**0.0**	0.7	0.1
20_2_1_40_A	18.0	0.7	1.5	**0.0**
20_2_1_40_B	**0.0**	**0.0**	0.1	**0.0**
20_2_1_80_A	60.0	68.7	0.9	**0.0**
20_2_1_80_B	11.8	**0.0**	0.3	**0.0**
20_2_3_40_A	6.8	0.9	0.6	**0.0**
20_2_3_40_B	**0.0**	**0.0**	0.8	**0.0**
20_2_3_80_A	23.2	86.0	0.6	**0.0**
20_2_3_80_B	7.4	**0.0**	0.1	**0.0**
20_4_1_20_A	3.5	0.1	2.4	0.6
20_4_1_20_B	0.5	**0.0**	2.9	0.1
20_4_1_40_A	178.6	68.3	6.9	**0.0**
20_4_1_40_B	6.7	0.1	1.7	**0.0**
20_4_3_20_A	1.2	**0.0**	4.7	0.1
20_4_3_20_B	1.3	**0.0**	3.5	0.7
20_4_3_40_A	380.8	90.2	28.2	**0.0**
20_4_3_40_B	47.2	0.5	6.4	**0.1**
40_4_1_40_A	1241.2	1321.5	4.5	**0.0**
40_4_1_40_B	65.6	0.3	1.0	**0.0**
40_4_1_80_A	74.6	255.6	8.6	**0.0**
40_4_1_80_B	353.6	175.7	0.3	**0.0**
40_4_3_40_A	243.9	49.3	3.3	**0.0**
40_4_3_40_B	55.2	1.5	1.0	**0.0**
40_4_3_80_A	115.2	244.3	1.8	**0.0**
40_4_3_80_B	60.1	282.4	0.7	**0.0**
All	94.5	82.7	3.0	**0.1**

Bold values indicate the best approach for the given family of instances

$\mathcal{M}_i$ of execution modes such that it meets the two following requirements:

- $\forall m, m' \in \mathcal{M}_i, q_{im} \neq q_{im'}$,
- $\forall m, m' \in \mathcal{M}_i, q_{im} < q_{im'} \rightarrow d_{im} > d_{im'}$.

Arbitrarily, we build $ov(\mathcal{I})$ considering that overlapping tasks are forbidden on the same turbine. Notice that, according to some experts in the field, it is reasonably realistic to only consider these subsets. After the generation of the tasks, we generate the set $\mathcal{R}$ of technicians using procedure **genTechnicians**().

Procedure genTasks

$\mathcal{I} \leftarrow \emptyset$
for $i \in \{1, \ldots, n_{\mathcal{I}}\}$ **do**
  ; /* Creation of a new task i */
  Associate randomly a wind turbine with task i by sampling the $Dn_{\mathcal{I}}^{\mathcal{W}}$ distribution
  Define the type $k \in K$ of the task according to the probabilities $p(k)$
  Define the impact of the task on the wind turbines by sampling the $Di_{impact}(k)$ distribution
  Draw randomly the skill s_i required by task i from the set $\mathcal{S}$
  Set the minimal (q_i^{MIN}) and the maximal (q_i^{MAX}) numbers of technicians that can perform task i during any given time period by sampling the $Di_{req}(k)$ $Dn_{\mathcal{I}}^{\mathcal{W}}$
  Generate a task duration d_i by sampling the $Di_{dur}(k)$ distribution
  $n_{\mathcal{M}_i} \leftarrow q_i^{MAX} - q_i^{MIN} + 1$
  $\mathcal{M}_i \leftarrow \emptyset$
  ; /* d_i^{prev}: duration of the last executing mode created for task i */
  for $m \in \{1, \ldots, n_{\mathcal{M}_i}\}$ **do**
    Create executing mode m for which task i requires $q_i^{\mathcal{M}}$ technicians and lasts $d_i^{\mathcal{M}}$ time periods with:
      $q_i^{\mathcal{M}} \leftarrow q^{MAX} - m + 1$
    ; /* We assume that the duration of a working day is 8 h. */
      $d_i^{\mathcal{M}} = \max(d_i^{prev} + 1, \lfloor \frac{d_i |\mathcal{T}|}{8 |\mathcal{D}| q_i^{\mathcal{M}}} + 0.5 \rfloor$
    Add the created executing mode to $\mathcal{M}_i$
    $d_i^{prev} \leftarrow d_i^{\mathcal{M}}$
  end
  Add the created task i to $\mathcal{I}$
end

Procedure genTechnicians

Let d^{unv} be the average number of time periods during which a technician is not available according to $Dr_{\#unv}$ and $Dr_{d^{unv}}$.
$\mathcal{R} \leftarrow \emptyset$
for $s \in \{1, \ldots, n_{\mathcal{S}}\}$ **do**
  ; /* compute the average total request of the tasks RS_s^{avg} */
  $RS_s^{avg} = \sum_{\substack{i \in \mathcal{I} \\ s_i = s}} \frac{1}{|\mathcal{M}_i|} \sum_{m \in \mathcal{M}_i} q_{im}$
  ; /* n_s minimum number of technicians mastering skill s */
  $n_s \leftarrow \epsilon \cdot \frac{RS_s^{avg}}{d^{unv}}$
  for $r \in \{1, \ldots, n_s\}$ **do**
    Create a technician mastering skills s and generate his or her unavailability time periods by sampling the $Dr_{\#unv}$ and $Dr_{d^{unv}}$ distributions
    Add this technician to $\mathcal{R}$
  end
end
for $r \in |\mathcal{R}|$ **do**
  Sample the $Dr_{\#skills}$ distribution to generate the number of skills mastered by technician r
  According to the previous value, generate additional skills for technician r
end

Table 14 Detail parameter setting used by the instance generator

Parameters	**Values**							
$(n_{\mathcal{T}}, n_{\mathcal{D}})$	$(10,5)$		$(20,5)$		$(20,10)$		$(40,10)$	
$n_{\mathcal{I}}$	20	40	20	40	40	80	40	80
$n_{\mathcal{S}}$	1 *or* 3							
$Dn^{\mathcal{L}}$	$\mathcal{U}(\{5,...,10\})$	$\mathcal{U}(\{8,...,12\})$	$\mathcal{U}(\{5,...,10\})$	$\mathcal{U}(\{8,...,12\})$	$\mathcal{U}(\{8,...,12\})$	$\mathcal{U}(\{12,...,20\})$	$\mathcal{U}(\{8,...,12\})$	$\mathcal{U}(\{12,...,20\})$
Dl^{xy}	$\mathcal{U}(\{0,...,80\}) \times \mathcal{U}(\{0,...,80\})$							
$Dn^{\mathcal{L}}_{\mathcal{W}}$	$\mathcal{U}(\{1,2,...,12\})$							
$Dn^{\mathcal{W}}_{\mathcal{I}}$	$\mathcal{U}(\{0,1,2\})$							
K	main maintenance tasks		inspection, gearbox oil change, other operations...					
$p(k)$	0.5		0.5					
$Di_{impact}(k)$	$\mathcal{U}(\{ST, BL\})$		$\mathcal{U}(\{NST, ST, BL\})$					
$Di_{dur}(k)$	$\mathcal{U}(\{20,40,60,80\})$		$\mathcal{U}(\{8,16\})$					
$Di_{req}(k)$	$2,3,4$		$2,3$					
$Dr_{\#skills}$	$\mathcal{U}(\{1,...,\lvert S\rvert\})$							
$Dr_{\mathbb{P}(unv)}$	0.25							
$Dr_{\#unv}$	1							
$Dr_{d^{unv}}$	$\mathcal{U}(\{1,...,2\})$							
ϵ	1 *or* 1.2							
Dw_{power}	$\mathcal{U}(\{2000, 2500, 3000\}$							
Δl^{min}	5							
Δr^{max}	25							
$\widehat{\phi}$	6							
Υ^{safety}_{max}	12							
Δl^{max}	15							
α	0.9							
δ	1		2		1		2	
$hours(t)$	$hours(t) = \begin{cases} 14 & \text{if } t \text{ is a rest time period} \\ 5^* & \text{if } t \text{ is not a rest time period and } \lvert\mathcal{T}\rvert/\lvert\mathcal{D}\rvert = 2 \\ 2.5^* & \text{if } t \text{ is not a rest time period and } \lvert\mathcal{T}\rvert/\lvert\mathcal{D}\rvert = 4 \end{cases}$							
$CF(\phi)$	piecewise linear approximation with the following 9 points							

ϕ	0	3.5	5.5	7	12.5	14	24.9	25	30
$CF(\phi)$	0	0	0.1	0.23	0.91	1	1	0	0

* We consider that the turbine is stopped slightly longer than the duration of a task (that has an impact on its execution) because the turbine has often to be stopped a bit earlier than the starting time of this task

NS No turbines are shut down during the execution of the task

ST The task impacts only the turbine on which it is executed. This turbine is stopped during the execution of the task (not during the rest time periods the task overlaps)

BL The task impacts only the turbine on which it is executed. This turbine is stopped during the execution of the task and during the rest time periods the task overlaps

Remark The model can also consider tasks that shut down multiple turbines in a wind farm (e.g., the maintenance of the wind farm substation), but this is extremely rare in practice

The last part of the generator concerns the parameters related to the objective function. For the sake of convenience, we introduce the set $\mathcal{T}^+$ of all time periods formed by the union of set $\mathcal{T}$ and the set of rest time periods that occur between each day. More specifically, we include a rest time period after every $\frac{|\mathcal{T}|}{|\mathcal{D}|}$ consecutive time periods of $\mathcal{T}$.

As it concerns the wind speed forecast at hub height, the main purpose is to use realistic values. First, we generate wind speed $\bar{\phi}^t_l$ for every location $l \in \mathcal{L}$ and every time period $t \in \mathcal{T}^+$ using a Rayleigh distribution with a scale parameter equal to $\widehat{\phi}\sqrt{\frac{2}{\pi}}$ (so that the expected wind speed is $\widehat{\phi}$). Since space correlation can be significant, we compute a corrected wind speed $\bar{\bar{\phi}}^t_l$ for every location l and every time period t as follows:

$$\bar{\bar{\phi}}^t_l = \frac{\sum_{\substack{l' \in \mathcal{L} \\ \text{s.t. } \Delta_{ll'} < \Delta l^{max}}} (\Delta l^{max} - \Delta_{ll'})\, \bar{\phi}^t_{l'}}{\sum_{\substack{l' \in \mathcal{L} \\ \text{s.t. } \Delta_{ll'} < \Delta l^{max}}} (\Delta l^{max} - \Delta_{ll'})}.$$

Wind speeds were generated independently from a time period to another one. However, this timewise independence assumption is unlikely to be verified in practice. To smooth out the speed values, we use a δ-weighted moving average that yields wind speed ϕ^t_l according to the following formula:

$$\phi^t_l = \frac{\bar{\bar{\phi}}^t_l + \sum_{t'=\max(0,t-\delta)}^{\max(0,t-1)} \alpha^{t-t'} \phi^{t'}_l}{1 + \sum_{t'=\max(0,t-\delta)}^{\max(0,t-1)} \alpha^{t-t'}}.$$

The resulting values are rounded to the nearest tenth. From our perspective, they compare well to realistic data.

Afterward, for each task $i \in \mathcal{I}$ and every time period $t \in \mathcal{T}$, we compute the binary parameter $\widetilde{\vartheta}^t_i$ equal to 1 if and only if $\phi^t_l < \Upsilon^{safety}_{max}$ (i.e., the task i can be scheduled during time period t according to safety concerns). Arbitrarily, we set each parameter ϑ^t_i equal to 1 for every task i and every time period t. We point out here that this choice makes the instances more complicated to solve as there is a wide

flexibility to schedule the maintenance operations. This also matches field observations.

The last step consists in computing the revenue value g_w^t for every wind turbine $w \in \mathcal{W}$ during each time period $t \in \mathcal{T}^+$. We compute the revenue from the nominal power P_w of the wind turbine and from the wind speed $\phi_{l_w}^t$. We also use an estimation $hours(t)$ of the number of hours during every time period t. More specifically, we compute the revenue g_w^t generated by each turbine $w \in \mathcal{W}$ that is available during time period $t \in \mathcal{T}$ as follows:

$$g_w^t = 0.08 \cdot P_w \cdot hours(t) \cdot CF(\phi_{l_w}^t).$$

where

- 0.08: is an approximation to the selling price in euros of 1 kWh of wind energy (this selling price is guaranteed for the next 10 years in France).
- $hours(t)$: estimation of the number of hours during time period $t \in \mathcal{T}^+$
- P_w: nominal power of wind turbine $w \in \mathcal{W}$
- $\phi_{l_w}^t$: wind speed forecast during time period t at the location of turbine $w \in \mathcal{W}$
- $CF(\phi)$: the ratio of the net electricity generated according to a wind speed equal to ϕ to the energy that could have been generated at full-power operation (this ratio is given by a piecewise linear function estimated from real data)

Finally, we compute a single postponing penalty across all tasks. This penalty is equal to the maximum loss of revenue that can be generated by the scheduling of a task of $\mathcal{I}$ plus one. With this definition, we almost always (if not always) ensure that postponing a task is non-profitable. With this penalty, we therefore almost ensure to schedule the maximum number of tasks according to the total number of technicians and their availability. This is quite in line with the practice in the field.

Table 14 presents the detail parameter setting used in the generation process.

References

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *Constraint-based scheduling: Applying constraint programming to scheduling problems*. Dordrecht: Kluwer.

Budai, G., Dekker, R., & Nicolai, R. (2008). Maintenance and production: A review of planning models. *Complex system maintenance handbook, Springer series in reliability engineering* (pp. 321–324). London: Springer.

Cordeau, J.-F., Laporte, G., Pasin, F., & Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, *13*(4), 393–409.

De Reyck, B., Demeulemeester, E., & Herroelen, W. (1998). Local search methods for the discrete time/resource trade-off problem in project networks. *Naval Research Logistics (NRL)*, *55*(6), 553–578.

Ding, F., Tian, Z., & Jin, T. (2013). Maintenance modeling and optimization for wind turbine systems: A review. In *2013 International conference on quality, reliability, risk, maintenance, and safety engineering (QR2MSE)*, pp. 569–575.

Froger, A., Gendreau, M., Mendoza, J., Pinson, E., & Rousseau, L.-M. (2016). Maintenance scheduling in the electricity industry: A literature review. *European Journal of Operational Research*, *251*(3), 695–706.

Kovács, A., Erds, G., Viharos, Z., & Monostori, L. (2011). A system for the detailed scheduling of wind farm maintenance. *CIRP Annals - Manufacturing Technology*, *60*(1), 497–501.

Malapert, A., Guéret, C., & Rousseau, L.-M. (2012). A constraint programming approach for a batch processing problem with non-identical job sizes. *European Journal of Operational Research*, *221*(3), 533–545.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research*, *34*(8), 2403–2435.

Pisinger, D., & Ropke, S. (2010). Large Neighborhood Search. In M. Gendreau & J.-Y. Potvin (Eds.), *Handbook of metaheuristics* (Vol. 146, pp. 399–419)., International series in operations research & management science New York: Springer.

Prud'homme, C., Fages, J.-G., & Lorca, X. (2014). *Choco3 documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S.

Rodriguez, J. (2007). A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological*, *41*(2), 231–245.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *Principles and Practice of Constraint Programming-CP*, *1998*, 417–431.