CHALMERS
UNIVERSITY OF TECHNOLOGY

revenue-based maintenance scheduling for wind turbines

Master's thesis in engineering mathematics and computational science

LUDVIG JAKOBSSON

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

revenue-based maintenance scheduling for wind turbines

LUDVIG JAKOBSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2023

revenue-based maintenance scheduling for wind turbines
LUDVIG JAKOBSSON

Supervisor: Ann-Brith Strömberg, Mathematical sciences
Examiner: Serik Sagitov, Mathematical Sciences

Master's Thesis 2023
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Revenue–based maintenance scheduling for wind turbines
LUDVIG JAKOBSSON
Department of mathematical sciences
Chalmers University of Technology

Abstract

The ability to effectively schedule maintenance so that lost revenue is minimized is an important aspect of wind turbine management, and a recent interest at the company Rabbalshede Kraft AB. The aim of this thesis was to formulate short–term maintenance scheduling models as integer linear optimization problems and to implement these using open–source software. Two models were formulated for the cases of in-house maintenance teams and hired external workers, respectively. These models take a wind forecast and spot prices as input to form the revenue forecast used in the optimization. Both models were implemented and the former was thoroughly tested in hypothetical maintenance situations. The results showed that a significant reduction in lost revenue is possible by solving for the optimal schedule. Further, the wind speed forecast and turbine performance were analysed with respect to the expected error in lost revenue.

Several methods of reducing run time of the optimization were studied. One of these, the so-called Dantzig–Wolfe decomposition, was formulated explicitly for both models and implemented for the in–house model. The result from a run time analysis showed that this method effectively reduces run time for a number of maintenance scheduling cases and allowed for practical solution of complex problems with free open–source software.

Keywords: wind energy, maintenance scheduling, integer optimization.

Acknowledgements

I would like to thank my supervisor Ann-Brith Strömberg for many helpful comments and insightful discussions. I would also like to thank Niklas Andersson at Rabbalshede Kraft AB for helping me plan this project and navigate the world of wind energy. A final thank you to my partner Corrynna for her support and for the cover image of this thesis.

Ludvig Jakobsson, Gothenburg, June 2023

Contents

List of Acronyms

Below are the acronyms used in this thesis, listed in alphabetical order.

CM	Corrective maintenance
D–RMP	Dual restricted master problem
ILP	Integer linear programming
LP	Linear programming
MAPE	Mean absolute percentage error
MP	Master problem
PC	Power curve
PM	Preventive maintenance
RMP	Restricted master problem

List of Figures

List of Tables

1

Introduction

1.1 Background

An important aspect of operating wind farms is performing maintenance on the various components in the turbines. Maintenance is associated with a number of costs including mobilisation cost, which is the cost of moving equipment to the site and paying the technicians' salaries, component cost, which is the cost of purchasing new components, and finally cost of losing out on production while the maintenance is being performed during which the turbine cannot produce electricity. All of these costs are subject to optimization in order to reduce the overall cost of operating the wind turbine. However, taking all of the perspectives into account would most probably form an optimization problem too complex to solve. By focusing on one of the above issues, we may define a solvable optimization problem which can give useful insight into the maintenance problem as a whole.

The company Rabbalshede Kraft AB (RKAB) manages around 150 wind turbines spread across 26 parks. With 32 employees and offices in Rabbalshede, Gothenburg, and Stockholm, RKAB utilizes both in–house technicians and other service providers in order to perform the required maintenance. Today, the company does not use any mathematical optimization models for maintenance scheduling but instead base these decisions on rough weather forecasts and professional experience.

A particular area of interest for RKAB is short-term maintenance scheduling in order to reduce the cost of losing out on production. This problem has been studied previously in for example [4] and solving for the optimal schedule has been shown to greatly contribute to reduced cost. The main idea in previous work has been to schedule maintenance when conditions for producing electricity are poor, most often due to low wind speeds, and to avoid maintenance when conditions are good. Another way to approach this, and one with more embedded information, is to consider when the potential revenue is good or poor. An important factor in good or poor revenue conditions, and one of interest to RKAB, is the selling-price of electricity, the so–called spot price. Consider a set of turbines I. The total revenue R_i^T of a turbine $i \in I$ for some discretized timespan with indices $t = 0 \dots T$ can be written as

$$R_i^T = \sum_{t=0}^{T} P_{it} \cdot e_t, \tag{1.1}$$

where e_t [euro/kWh] is the spot price during time step t and P_{it} [kWh] is the electricity produced by turbine i during time step t. Electricity prices e_t have a similar effect on maintenance scheduling as wind speeds, namely that maintenance should be scheduled when prices are low and avoided when prices are high. The effect of taking both wind speed and prices into account in an optimization model is not well understood but may offer further reductions to costs.

Hourly spot prices for the next day are published every day at 13:00 for all of Sweden's four electricity zones. These prices, together with the weather forecast, give a prognosis for the revenue of the next day and also when during the day the most revenue will be produced. The main issue with using these prices in a maintenance scheduling workflow is the short time span between spot prices being published and maintenance being carried out. It becomes a practical challenge to reschedule maintenance between 13:00 and the end of the workday, and to also communicate the new schedule to relevant staff, e.g. technicians. In some cases it may even be impossible to plan new maintenance jobs due to inflexible service contracts with external service providers. This work may serve as an incentive for developing more flexible contracts, should reduced costs be shown.

The practical issues above may be mitigated by scheduling maintenance further into the future than just one day. This, however, requires spot prices to be approximated for the hours after the next day, when no fixed prices are available. The ideal approximation would be hourly forecasted spot prices but since these may not be very accurate, an average spot price may also be used for the scheduling period extending past the next day.

1.2 Aim

The aim of this project is to formulate and implement an optimization model for short-term scheduling of maintenance jobs. In this case, short-term implies that the scheduling horizon is no longer than a week. The model should take both wind speeds and electricity prices into account as well as any required constraints such as no maintenance being carried out at night. Using forecast data on wind speeds and hourly spot prices, an optimal solution to the model should propose a schedule for the given maintenance jobs with the smallest expected lost revenue.

The model will be implemented using the commercial solver Gurobi for testing as well as the open–source solver GLPK for the final implementation. The aim is that RKAB should have all the tools necessary for running the optimization without needing to purchase any software.

A second aim of the project is to study the effect of including spot prices by measuring the difference in cost reduction from a model utilizing both wind and prices and a model utilizing only wind speeds.

1.3 Limitations

There are several aspects of the larger problem of maintenance scheduling which will not be dealt with in this thesis. First, no finished application for maintenance scheduling will be developed during the 20 weeks of this project. Furthermore, while the issue with inflexible service contracts is important and will affect the usability of the model, no direct attempts at fixing these issues will be made. The project and its results will instead serve as incentive for more flexible contracts.

There are many factors which contribute to how good or bad the revenue conditions are at any given time. These include wind speed and spot price but also wind density, wind direction, temperature, etc. This project will not take into account all of these factors but will instead focus on wind speed and spot price. This means that the parameter P_{it} in equation (1.1) only depends on the wind speed and the specific turbine i, all other factors being constant. This is common in wind turbine analysis where P_{it} is often represented by the so called power curve. Finally it is not within the scope of this project to fully verify the results through actual maintenance scheduling and execution since this would most probably take too much time to do.

2

Revenue forecasting in wind energy

2.1 Power curve

Every type of wind turbine has specified power levels, measured in Watts, for each wind speed from 0 m/s to 20 m/s. To get the power levels, other factors which contribute to production, such as air density, are set to some standard value thereby making power production depend only on wind speed. Using linear interpolation between the given power levels and writing power as a function of wind speed for a given wind turbine, we get the power function $P(w)$, i.e. the expected momentary power produced when the wind speed is w m/s. The curve produced when plotting $P(w)$ for $w \in [0, 20]$ is commonly called the power curve (PC) of the turbine. The main use of the PC is judging the performance of the turbine by comparing the resulting PC over some time interval to the default PC specified by the manufacturer. Figure 2.1 shows the default and the resulting PC for one RKAB turbine during January 2023. It is evident that the wind speed never reached above 15 m/s and that the turbine performed above its specified levels for most of the wind speeds, while it did not reach the maximum level of the default PC.

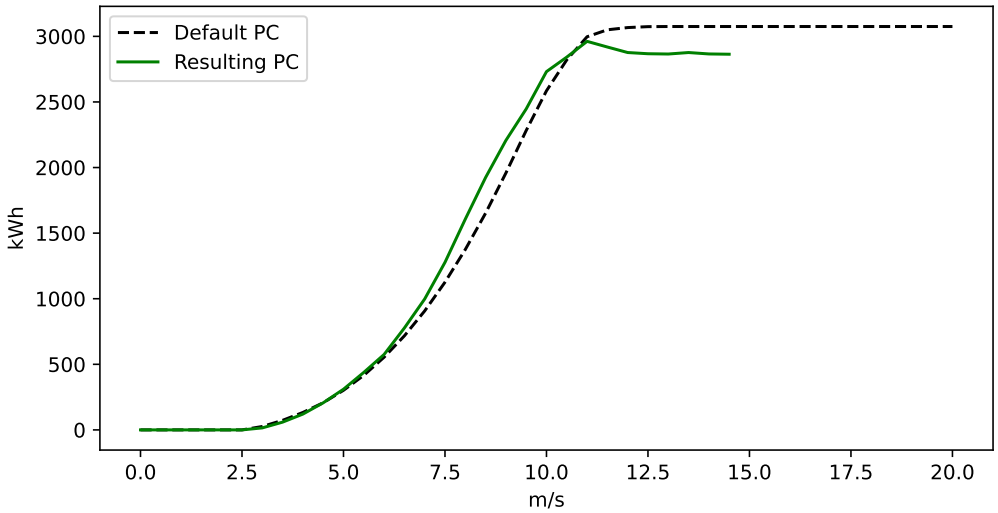

Figure 2.1: *The default power curve along with the resulting power curve for a Vestas V112 3.0MW wind turbine during January 2023*

By using a wind speed forecast $\{w_t\}_{t=0}^{T}$ for some time steps $t = 0, \dots T$ such that the length of each time step is τ, we get the expected power production for each time step as $\tau P(w_t)$. If τ is equal to one hour, $\tau P(w_t) = P(w_t)$ gets the standard unit of power over time, Watt-hours. This yields a simple power forecast which can be used to evaluate the expected power output at different times. This is fundamental in optimal maintenance scheduling. For the state of the art in power forecasting methods, see [7].

2.2 Spot prices

In northern Europe, the price at which energy companies sell and purchase electricity, the so called spot price, is set on the power exchange Nord Pool, [1]. Supply and demand is reflected by submitted orders for buying or selling electricity and a market price is calculated accordingly. Spot prices emerging from the power exchange depend on a multitude of sources related to supply and demand. Heavy rain fall and strong winds drive up supply, thus lowering the price, while cold weather increases the need for heating thus driving up demand and price. The price of other fuels such as natural gas also contributes to the spot price. For these reasons, forecasting of prices is in general difficult.

Spot prices for the next day are set and made available at 13:00 every day. These prices, together with the simple power forecast described above, yield an hourly revenue forecast defined as

$$R_t = \tau P(w_t) e_t, \quad t = 0, \dots, T \tag{2.1}$$

The problem of scheduling maintenance such that the lost revenue is minimized can thus be solved by planning maintenance jobs according to the revenue forecast.

2.3 Forecasting spot prices

As mentioned above, spot prices are set and made available for one day at a time and forecasting of prices is difficult. In [8], success has been had in forecasting weekly average spot prices using historic prices and data on water reservoir levels and inflow. It may be possible to use these weekly averages as constant electricity prices in place of hourly prices. Recent advances have also been made using probabilistic forecasting, [10]. Finally, the company Rebase, [3], which has supplied wind forecasts for this project, are also working on a spot price forecast using market analysis techniques.

2.4 Revenue forecasting errors

There are multiple sources of error in the revenue forecast described above. Figure 2.1 shows the first, in which the turbine did not produce power according to the

manufacturers specification given the wind speed. The second source of error is in the wind speed forecast $\{w_t\}_{t=0}^{T}$. The Swedish Meteorological and Hydrological Institute, SMHI, defines a correct wind speed forecast as one with an absolute error less than 2 m/s. According to their data on forecast follow-up, [2], the percentage of correct same-day wind speed forecasts, measured from 08:00 to 20:00, in January 2023 was 80.4%. The percentage dropped to 72.1% and 58.4% for 3-day and 5-day forecasts, respectively. When using a spot price forecast, a third source of error is added to the revenue forecast.

3

Mathematical optimization theory

3.1 Integer programming

Unlike linear programming (LP), in which variables are continuous, integer linear programming (ILP) features variables which are restricted to whole numbers. This may seem like a simplification of LP since there are fewer feasible solution to consider, however, since functions defined on a discrete set of points lack well defined gradients, ILP problems are in fact much more difficult to solve than their LP counterparts. Consider the LP problem and its integer restriction in the model (3.1).

$$\begin{aligned} z^*_{\text{LP}} = \max\ & 2x_1 + 3x_2 \\ \text{s.t. } & x_1 + \ x_2 \leq 3 \\ & x_1 + 3x_2 \leq 6 \\ & x_1, \quad x_2 \geq 0 \end{aligned} \qquad \begin{aligned} z^*_{\text{ILP}} = \max\ & 2x_1 + 3x_2 \\ \text{s.t. } & x_1 + \ x_2 \leq 3 \\ & x_1 + 3x_2 \leq 6 \\ & x_1, \quad x_2 \in \mathbb{Z}_+ \end{aligned} \tag{3.1}$$

The feasible sets and optimal solutions of these problems are shown in Figure 3.1. For the LP problem, the optimal solution is $x^*_{\text{LP}} = (1.5, 1.5)^T$ with optimal objective value $z^*_{\text{LP}} = 7.5$. Since x^*_{LP} is not integer, it cannot coincide with the solution to the restricted problem. In the integer case, the solution is $x^*_{\text{ILP}} = (2, 1)^T$ with optimal objective value $z^*_{\text{ILP}} = 7$. Note that $z^*_{\text{ILP}} \leq z^*_{\text{LP}}$ which is a general result for any restriction of the feasible set to a maximization problem. For a minimization problem, the reverse inequality holds.

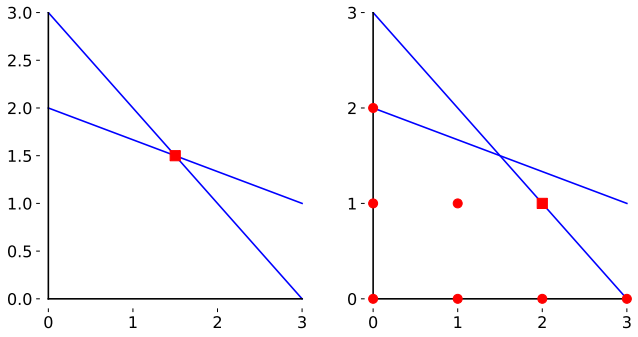

Figure 3.1: *The feasible sets and optimal solutions of the problems in equation* (3.1)*. The optimal solutions are marked by red squares and in the ILP problem (right) the feasible set is marked by red circles.*

The restricted problem in (3.1) can easily be solved by calculating the objective value in each feasible point and comparing. However, for problems with large feasible sets, this strategy is not viable. The reason integer problems are more difficult to solve than LP problems is that in the LP case, an optimal solution can always be found in an extreme point. This property is used in the popular simplex algorithm to efficiently solve any LP problem. In the integer case, no such property exists, see Figure 3.1, and so the Simplex algorithm does not work. Thus, other methods must be applied to solve general ILP problems.

3.2 Methods for solving ILP problems

Several general methods exist for solving ILP problems. The simplest, enumeration, consists of calculating the objective value in each feasible point and comparing. This method was used to solve the problem above. As problems grow in size, enumeration quickly becomes too slow for practical use. One method for faster solution of ILPs is the widely used Branch and Bound method, also known as implicit enumeration. The general idea of Branch–and–Bound is to only examine points which may give an improvement to the best objective value found so far. This involves solving relaxed subproblems to the original problem, often LP relaxations, in smaller and smaller feasible regions through branching. In a 0/1 ILP problem, in which variables take values in the set $\{0, 1\}$, branching involves fixing one variable to 0 in the left branch, and to 1 in the right. A relaxation of the original problem is then solved in each branch with the branching variable fixed to the respective value. Through this, both optimistic and pessimistic bounds on the optimal objective value are found which converge to the optimal value. For an in-depth introduction to Branch and Bound, see [9, Chapter 15].

In the ILP problem in (3.1), the Branch–and–Bound algorithm would start in the optimal solution to the LP relaxation. A possible branching would then be to consider $x_1 \leq 1$ and $x_1 \geq 2$, respectively, in which case the optimal solution would be found in the latter case. Branch and Bound is used in most open source and commercial solvers and works well for solving quite large integer linear programming problems, however, further improvements to solution times can be made using other methods or by combining methods.

3.2.1 Column generation

One way to improve efficiency when solving linear programs, in particular when the number of variables is large, is to make use of column generation methods. Here, columns refer to the variables and all of their information, i.e. cost and constraint coefficients. The idea behind column generation is to not consider all columns at the same time but instead iteratively generate the ones that lead to an improved objective value. To this end, a master problem containing all columns is formulated and then restricted to only a few starting columns. These starting columns must be feasible solutions to the original problem and the efficiency of the algorithm depends heavily on the specific choice of columns. Next, in order to find which columns to

add in each iteration, a subproblem is solved and the so-called reduced costs is calculated for each possible new column, similarly to the simplex algorithm. The specific properties of the subproblems are problem dependent but often a decomposition is made so that subproblems are either easy to solve by some general method or so that specialized and efficient algorithms exist. After the subproblems have been solved, a column is added to the restricted master problem which, when solved, gives information for the next iteration.

A common decomposition method used in column generation is the Dantzig–Wolfe decomposition, [6]. Consider the linear minimization problem (3.2),

$$z^* := \min_x \quad c^T x \tag{3.2a}$$
$$\text{s.t.} \quad Ax \geq b \tag{3.2b}$$
$$x \in X_{\text{LP}}, \tag{3.2c}$$

where $X_{\text{LP}} := \{x \in \mathbb{R}_+ \mid Dx \geq d\}$ for some valid constraints $Dx \geq d$. Assume that the constraints $Ax \geq b$ are complicating, meaning they should not be included in the subproblem. The set X_{LP} can be written as the convex hull of it's extreme points, i.e. $\text{conv}_{q \in Q_{\text{LP}}}\{x^q\}$, where x^q is an extreme point to X_{LP} and Q_{LP} is the index set of all extreme points. By utilizing the fact that every point in X_{LP} can be written as a convex combination of its extreme points, the problem (3.2) can now be rewritten with the convex combination coefficients λ_q, yielding the master problem (MP) of the Dantzig–Wolfe decomposition, as

$$z^* = z^*_{\text{MP}} := \min_\lambda \quad \sum_{q \in Q_{\text{LP}}} c^T x^q \lambda_q \tag{3.3a}$$
$$\text{s.t.} \quad \sum_{q \in Q_{\text{LP}}} Ax^q \lambda_q \geq b \tag{3.3b}$$
$$\sum_{q \in Q_{\text{LP}}} \lambda_q = 1 \tag{3.3c}$$
$$\lambda_q \geq 0, \quad q \in Q_{\text{LP}}. \tag{3.3d}$$

The MP is then restricted to a subset of extreme points (columns) with the index set $\bar{Q}_{\text{LP}} \subset Q_{\text{LP}}$ which yields the restricted master problem (RMP)

$$z^*_{\text{MP}} \leq z^*_{\text{RMP}} := \min_\lambda \quad \sum_{q \in \bar{Q}_{\text{LP}}} c^T x^q \lambda_q \tag{3.4a}$$
$$\text{s.t.} \quad \sum_{q \in \bar{Q}_{\text{LP}}} Ax^q \lambda_q \geq b \tag{3.4b}$$
$$\sum_{q \in \bar{Q}_{\text{LP}}} \lambda_q = 1 \tag{3.4c}$$
$$\lambda_q \geq 0, \quad q \in \bar{Q}_{\text{LP}}. \tag{3.4d}$$

Note that the optimal objective value of (3.4) a greater than that of the full master problem (3.3) since it is restricted to only a subset of all feasible solutions to the model (3.2). The optimal objective values of the above problems are related as

$$z^* = z^*_{\text{MP}} \leq z^*_{\text{RMP}}. \tag{3.5}$$

The next step of the decomposition is to consider the linear programming dual problem of the RMP (D–RMP) which is formulated as.

$$z^*_{\text{RMP}} = \max_{u,w} \quad b^T u + w \tag{3.6a}$$
$$\text{s.t.} \quad (Ax^q)^T u \leq c^T x^q, \quad q \in \bar{Q}_{\text{LP}} \tag{3.6b}$$
$$u \geq 0 \tag{3.6c}$$
$$w \in \mathbb{R}. \tag{3.6d}$$

Solving problem (3.6) yields the solution $(\bar{u}, \bar{w})$ which in turn defines the column generation subproblem

$$\bar{c}_q := \min_{q \in Q_{\text{LP}}} \left\{ c^T x^q - \bar{u}^T A x^q - \bar{w} \right\}. \tag{3.7}$$

Any solution to (3.7) with reduced cost $\bar{c}_q < 0$ is a candidate for a new column in the RMP. By iterating through this procedure, one recovers an optimal solution to the LP problem as the convex combination $\sum_{q \in \bar{Q}_{\text{LP}}} \lambda_q x^q$ of the columns generated when $\bar{c}_q \geq 0$ for all $q \in Q_{\text{LP}}$.

3.2.2 Branch and Price

The column generation procedure described above is a method for solving LP problems and can thus be used to solve the LP relaxation of an ILP problem. By combining column generation with Branch and Bound, large ILP problems can be solved much more efficiently than with Branch and Bound alone. One issue to consider when combining these methods is that the columns included in the optimal solution to the relaxed problem may not be enough to solve the integer problem. Thus, the column generating procedure must be performed for every new branch. An algorithm which does just this is called Branch and Price. See Figure 3.2 for a flowchart describing the Branch and Price algorithm. Note that the flowchart describes the solution of a minimization problem, however, switching the inequalities for the reduced costs after the subproblem stage makes the process valid for maximization problems. For a full definition of Dantzig–Wolfe decomposition and Branch and Price, as well as other decomposition methods, see [5, Chapter 8].

Another way to use the Dantzig–Wolfe decomposition to solve ILP problems is to directly solve the subproblem with integer requirements. This approach avoids the involved implementation of Branch and Price but may not be as efficient in terms of run time since an ILP solver is required to solve the subproblem. The decomposition in Chapter 4 uses this approach in place of Branch and Price.

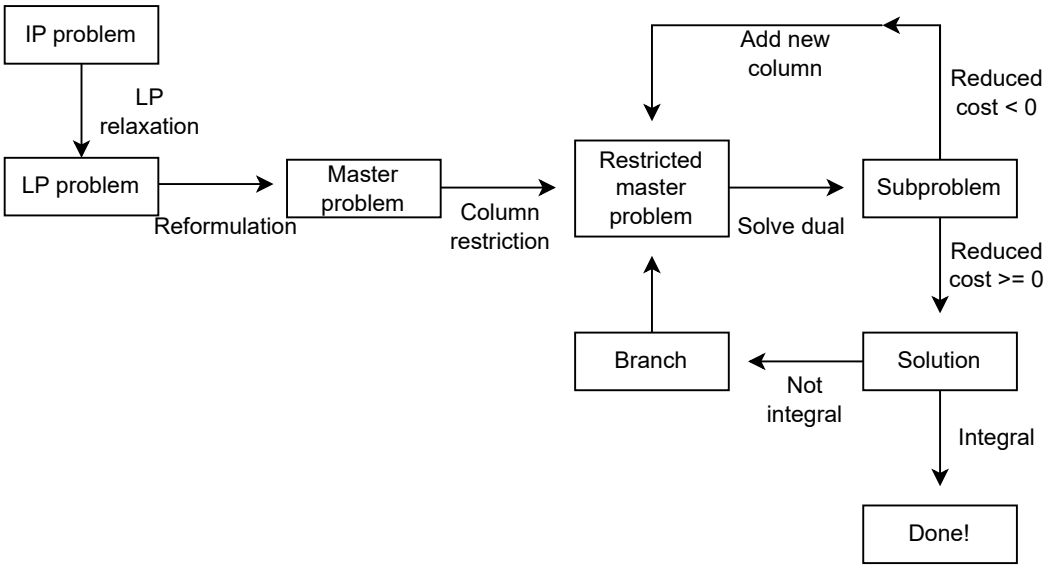

Figure 3.2: *Flowchart showing the solution process in the Branch and Price algorithm for solving integer linear programs under minimization.*

4

Integer linear programming models of the short-term maintenance scheduling problem

In this chapter, two maintenance scheduling models are presented. In the first model, maintenance is only performed by permanent in-house staff without extra costs incurred, while in the second model, temporary workers are available for hire which adds a cost to maintenance.

4.1 In-house model

The case of maintenance scheduling using in–house staff can be described as follows. A company has one wind park with a set of turbines I. Each turbine $i \in I$ requires some maintenance, with the expected maintenance time p_i. The scheduling time period is discretized with time steps $t = 0, \ldots, T$, for some $T \geq 0$, where the time between each step is τ hours. A set J contains all available maintenance teams at the company. These are all assumed to be permanent teams meaning that they do not carry any costs except for a monthly salary. For each step $t = 0, \ldots, T$, the parameter $b_{jt} \in \{0, 1\}$ is 1 if and only if team $j \in J$ is available for work during time step t. Thus, for each team $j \in J$, this parameter defines the daily schedule of team j. The turbines are divided into two subsets depending on the state of each turbine. The first subset I_{PM} contains turbines which are running but require preventive maintenance meaning that the turbine has to be stopped for the duration of the maintenance. The second subset I_{CM} contains turbines requiring corrective maintenance which means that the turbine is stopped from the start of the scheduling period, i.e. $t = 0$, and can only be started once maintenance is finished.

The objective is to schedule maintenance of all turbines such that the total lost revenue during the scheduling period is minimized. To this end, a wind speed forecast as well as a spot price forecast are utilized. The wind speed forecast is used in the power function $P(w)$ from Chapter 2.1 to estimate the power output during each time step. Together with the price forecast, this yields the revenue forecast fore the scheduling period. The problem is formulated as an integer linear optimization problem below.

4.1.1 Sets

The following sets are included in the model.

- $\mathcal{T}$ – set of all time steps
- I – set of all turbines
- I_{PM} – set of turbines requiring PM
- I_{CM} – set of turbines requiring CM
- J – set of permanent maintenance teams

4.1.2 Parameters

The following parameters are included in the model.

- $T \in \mathbb{Z}_+$ – the last time step in the scheduling time period
- $\tau \in \mathbb{R}_+$ – hours between two consecutive time steps
- $w_{it} \in \mathbb{R}_+$ – wind speed, m/s, at turbine $i \in I$ during time step $t \in \mathcal{T}$
- $e_t \in \mathbb{R}_+$ – spot price, €/kWh, during time step $t \in \mathcal{T}$
- $P_i(w) \in \mathbb{R}_+$ – power produced, kW, by turbine $i \in I$ at wind speed w
- $R_{it} = \tau P_i(w_t) e_t$ – revenue forecast, €, from turbine $i \in I$ during time step $t \in \mathcal{T}$
- $p_i \in \mathbb{Z}_+$ – duration, in hours, to complete the maintenance task at turbine $i \in I$
- $b_{jt} \in \{0, 1\}$ – 1 if and only if team $j \in J$ is available during time step $t \in \mathcal{T}$
- $C \in \mathbb{R}_+$ – penalty added per unfinished job on turbines $i \in I_{\mathrm{PM}}$

4.1.3 Variables

The model contains three sets of binary variables. The decision variables

$$x_{ijt} = \begin{cases} 1, & \text{if team } j \in J \text{ starts maintenance on turbine } i \in I \text{ at the} \\ & \text{beginning of time step } t \in \mathcal{T}, \\ 0, & \text{otherwise,} \end{cases}$$

and the auxiliary variables

$$y_{it} = \begin{cases} 1, & \text{if turbine } i \in I \text{ is not running during time step } t \in \mathcal{T}, \\ 0, & \text{otherwise;} \end{cases}$$

$$s_i = \begin{cases} 1, & \text{if maintenance on turbine } i \in I \text{ is not performed,} \\ 0, & \text{otherwise.} \end{cases}$$

4.1.4 Model formulation

The model is formulated in (4.1). Note the set

$$\mathcal{T}_i^p := \{0, \ldots, T - p_i + 1\},$$

which is defined to make the notation shorter.

$$z^* = \min \quad \sum_{i \in I} [C s_i + \sum_{t=0}^{T} R_{it} y_{it}] \tag{4.1a}$$

$$\text{s.t.} \quad x_{ijt} \leq y_{is}, \qquad s = t, \ldots, t + p_i - 1,\ t \in \mathcal{T}_i^p,\ i \in I_{\text{PM}},\ j \in J \tag{4.1b}$$

$$x_{ijt} \leq y_{is}, \qquad s = 0, \ldots, t + p_i - 1,\ t \in \mathcal{T}_i^p,\ i \in I_{\text{CM}},\ j \in J \tag{4.1c}$$

$$\sum_{j \in J} \sum_{t=0}^{T-p_i+1} x_{ijt} + s_i = 1, \qquad i \in I \tag{4.1d}$$

$$s_i = 0, \qquad i \in I_{\text{CM}} \tag{4.1e}$$

$$\sum_{i \in I} \sum_{s=\max\{0, t-p_i+1\}}^{t} x_{ijs} \leq b_{jt}, \qquad t \in \mathcal{T},\ j \in J \tag{4.1f}$$

$$x_{ijt} \in \{0, 1\}, \qquad t \in \mathcal{T},\ i \in I,\ j \in J \tag{4.1g}$$

$$y_{it} \in \{0, 1\}, \qquad t \in \mathcal{T},\ i \in I \tag{4.1h}$$

$$s_i \in \{0, 1\}, \qquad i \in I. \tag{4.1i}$$

Constraints (4.1b) state that turbine i has to be stopped from the start of a preventive maintenance until p_i time steps later. Constraints (4.1c) state that, in the case of corrective maintenance, the turbine is stopped from the start of the scheduling period until the maintenance is finished. Constraints (4.1d) together with (4.1e) make sure that if preventive maintenance is not performed on a turbine, a penalty C is added to the objective, and also, that each performed job is completed before there is no time left in the day and that one team is responsible for the complete maintenance job. Note that the penalty added for unfinished jobs is purely for modelling purposes and should not be considered as an actual cost in practical applications. Finally, constraints (4.1f) ensure that every team has to be available for the duration of their respective assigned maintenance jobs.

4.2 In–house and external workers

In cases where the permanent maintenance teams are unable to keep up with the jobs, additional reduction in lost revenue may be achieved by hiring temporary workers to perform maintenance. Consider a set of wind turbines owned and maintained by the wind farm manager. As in Chapter 4.1, there is a number of permanent teams available at each time step which do not carry an additional cost. It is now, however, also possible to hire temporary maintenance teams to perform jobs with an added cost to the objective function. It is clear that the hiring of a team must be motivated by a reduction in lost revenue which is greater than the cost of hiring the team.

Consider a partition of the set J into the set of permanent workers J_{perm} and the set of possible temporary workers J_{temp}. Since the number of temporary teams hired is to be optimized, this needs to be represented using a new variable z_{jt}. Note also that the parameter b_{jt} now defines the time steps when temporary teams are available for hire for all $j \in J_{\text{temp}}$. Hiring a temporary team must be associated with an additional cost, so a new parameter $c_t \in \mathbb{R}_+$ is also added. These new sets, parameters, and variables are defined as follows

- J – set of all maintenance teams,
- J_{perm} – set of all permanent maintenance teams,
- J_{temp} – set of all possible temporary maintenance teams,
- $c_t \in \mathbb{R}_+$ – cost of hiring one temporary maintenance team during time step $t \in \mathcal{T}$.

$$z_{jt} = \begin{cases} 1, & \text{if team } j \in J_{\text{temp}} \text{ is hired during time step } t \in \mathcal{T}, \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

A new integer linear programming model is defined in (4.3)

$$z^* = \min \quad \sum_{t=0}^{T} \Big[\sum_{i \in I} R_{it} y_{it} + \sum_{j \in J_{\text{temp}}} c_t z_{jt} \Big] + \sum_{i \in I} C s_i \tag{4.3a}$$

$$\text{s.t.} \quad x_{ijt} \le y_{is}, \qquad s = t, \dots, t + p_i - 1,\ t \in \mathcal{T}_i^p,\ i \in I_{\text{PM}},\ j \in J \tag{4.3b}$$

$$x_{ijt} \le y_{is}, \qquad s = 0, \dots, t + p_i - 1,\ t \in \mathcal{T}_i^p,\ i \in I_{\text{CM}},\ j \in J \tag{4.3c}$$

$$\sum_{j \in J} \sum_{t=0}^{T-p_i+1} x_{ijt} + s_i = 1, \qquad i \in I \tag{4.3d}$$

$$s_i = 0, \qquad i \in I_{\text{CM}} \tag{4.3e}$$

$$\sum_{i \in I} \sum_{s=\max\{0, t-p_i+1\}}^{t} x_{ijs} \le b_{jt}, \qquad t \in \mathcal{T},\ j \in J_{\text{perm}} \tag{4.3f}$$

$$\sum_{i \in I} \sum_{s=\max\{0, t-p_i+1\}}^{t} x_{ijs} \le z_{jt}, \qquad t \in \mathcal{T},\ j \in J_{\text{temp}} \tag{4.3g}$$

$$z_{jt} \le b_{jt}, \qquad t \in \mathcal{T},\ j \in J_{\text{temp}} \tag{4.3h}$$

$$x_{ijt} \in \{0, 1\}, \quad t \in \mathcal{T},\ i \in I,\ j \in J \tag{4.3i}$$

$$y_{it} \in \{0, 1\}, \quad t \in \mathcal{T},\ i \in I \tag{4.3j}$$

$$z_{jt} \in \{0, 1\}, \quad t \in \mathcal{T},\ j \in J_{\text{temp}} \tag{4.3k}$$

$$s_i \in \{0, 1\}, \quad i \in I. \tag{4.3l}$$

In the objective function (4.3), a cost for temporary teams is added for every time step $t \in \mathcal{T}$. Additionally, the constraints (4.1f) have been divided into the two sets of constraints (4.3f) and (4.3g) by the partitioning of J into J_{perm} and J_{temp}.

Recall that these constraints ensure that a maintenance team has to be available for the entire duration of the maintenance jobs they are undertaking. Since z_{jt} is a variable, this model now allows for purchasing extra maintenance capacity if beneficial. In order to specify at which times temporary workers are available for hire, the constraints (4.3h) are added for each $j \in J_{\text{temp}}$ which restrict the variables z_{jt} to be 0 for time steps t when $b_{jt} = 0$.

4.3 Improving run times

In order to achieve faster run times for the two models presented above, two actions can be taken. First, the integer requirements on the variables y_{it} and s_i can be relaxed to non-negative real numbers for both models. This is possible because of the structure of the constraints containing these variables. Consider the in-house model (4.1). Constraints (4.1b) and (4.1c) ensure that y_{it} only takes values in the set $\{0, 1\}$ since x_{ijt} is binary and y_{it} is minimized in the objective function. Meanwhile, the constraints (4.1d) are equality constraints, thus ensuring that

$$s_i = \begin{cases} 0, & \text{if } x_{ijt} = 1 \text{ for some } j \in J \text{ and } t = 0, \dots, T - p_i + 1, \\ 1, & \text{otherwise.} \end{cases} \tag{4.4}$$

Since no other cases exist, s can be relaxed. In the case of the model (4.3), the variable z can be relaxed to be continuous on the interval $[0, 1]$. The reason for removing integer requirements on some variables is that in the Branch and Bound method discussed in Chapter 3.2, one only needs to perform branching on integer variables since the rest will be feasible in solutions in the LP relaxation of the problem. This reduces the number of branches required to solve the problem, thus improving the run time.

Further increases in speed can be achieved through column generation and a decomposition as described in Chapter 3.2. The following section describes the proposed Dantzig–Wolfe decompositions for the two models presented above. Note that the Branch and Price algorithm is not implemented for these models. Instead, the subproblem is formulated as an ILP problem so that all columns generated are integer.

4.3.1 Dantzig–Wolfe decomposition

In order to successfully reduce run time by column generation, the subproblem (3.7) must be easy to solve. For the in-house model above, this can be done by making sure that the subproblem separates over $i \in I$ into $|I|$ independent and smaller subproblems, meaning they can be solved in parallel. To enable this, the following sets are defined,

- $B_0 = \big\{(j, t) \in J \times \mathcal{T} \,|\, b_{jt} = 0\big\}$,
- $B_1 = \big\{(j, t) \in J \times \mathcal{T} \,|\, b_{jt} = 1\big\}$.

The constraints (4.1f) can now be separated into the sets B_0 and B_1 as follows,

$$x_{ijs} = 0, \quad s = \max\{0, t - p_i + 1\}, \ldots, t,\ i \in I,\ (j,t) \in B_0, \tag{4.5}$$

$$\sum_{i \in I} \sum_{s=\max\{0,t-p_i+1\}}^{t} x_{ijs} \leq 1, \quad (j,t) \in B_1, \tag{4.6}$$

where the inequality turns into an equality in (4.5) since x has binary requirements. By letting the constraints (4.6) remain in the Dantzig–Wolfe master problem (3.3) and letting the constraints (4.5) be part of the subproblem together with the remaining constraints of the model (4.1), a Dantzig–Wolfe decomposition with subproblems separated over I is achieved.

Let V be the set of feasible points in model (4.1) with the constraints (4.6) relaxed, that is, the set of points $v := (x, y, s)$ such that the following inequalities hold:

$$\begin{aligned}
x_{ijt} &\leq y_{is}, \quad s = t, \ldots, t + p_i - 1,\ t \in \mathcal{T}_i^p,\ i \in I_{\mathrm{PM}},\ j \in J \\
x_{ijt} &\leq y_{is}, \quad s = 0, \ldots, t + p_i - 1,\ t \in \mathcal{T}_i^p,\ i \in I_{\mathrm{CM}},\ j \in J \\
\sum_{j \in J} \sum_{t=0}^{T-p_i+1} x_{ijt} + s_i &= 1, \quad i \in I \\
s_i &= 0, \quad i \in I_{\mathrm{CM}} \\
x_{ijs} &= 0, \quad s = \max\{0, t - p_i + 1\}, \ldots, t,\ i \in I,\ (j,t) \in B_0 \\
x_{ijt} &\in \{0,1\}, \quad t \in \mathcal{T},\ i \in I,\ j \in J \\
y_{it} &\in \{0,1\}, \quad t \in \mathcal{T},\ i \in I \\
s_i &\in \{0,1\}, \quad i \in I.
\end{aligned}$$

Let $v^q = (x^q, y^q, s^q)$ be an extreme point to the polyhedron $\mathrm{conv}\{V\}$ and let Q be the index set of all the extreme points. The problem (4.1) can now be rewritten using a convex combination of extreme points. This yields the master problem (MP) of the Dantzig–Wolfe decomposition (4.7).

$$z^* \geq z^*_{\mathrm{MP}} := \min_{\lambda} \quad \sum_{i \in I} \sum_{q \in Q^i} [C s_i^q + \sum_{t=0}^{T} R_{it} y_{it}^q] \lambda_{iq} \tag{4.7}$$

$$\begin{aligned}
\text{s.t.} \quad \sum_{i \in I} \sum_{q \in Q^i} \sum_{s=\max\{0,t-p_i+1\}}^{t} x_{ijs}^q \lambda_{iq} &\leq 1, \quad (j,t) \in B_1 \\
\sum_{q \in Q^i} \lambda_{iq} &= 1, \quad i \in I \\
\lambda_{iq} &\geq 0, \quad i \in I,\ q \in Q^i.
\end{aligned}$$

The sets Q^i above are the index sets of extreme points for each turbine $i \in I$. This partition of Q has been made so that the subproblem will separate over I. Since the number $|Q|$ of variables (columns) in the MP is large, the problem should be restricted to a small subset of columns $\bar{Q} \subseteq Q$. This yields the restricted master

problem (RMP) (4.8), defined as

$$z^*_{\text{MP}} \leq z^*_{\text{RMP}} := \min_{\lambda} \quad \sum_{i\in I}\sum_{q\in\bar{Q}^i} [Cs^q_i + \sum_{t=0}^{T} R_{it}y^q_{it}]\lambda_{iq} \tag{4.8}$$

$$\text{s.t.} \quad \sum_{i\in I}\sum_{q\in\bar{Q}^i} \sum_{s=\max\{0,t-p_i+1\}}^{t} x^q_{ijs}\lambda_{iq} \leq 1, \quad (j,t) \in B_1$$

$$\sum_{q\in\bar{Q}^i} \lambda_{iq} = 1, \quad i \in I$$

$$\lambda_{iq} \geq 0, \quad i \in I,\ q \in \bar{Q}^i.$$

Next, the linear dual to the restricted master problem, D–RMP, is formed as

$$(\bar{u},\bar{w}) = \arg\max \quad \sum_{(j,t)\in B_1} u_{jt} + \sum_{i\in I} w_i \tag{4.9}$$

$$\text{s.t.} \quad \sum_{(j,t)\in B_1} \sum_{s=\max\{0,t-p_i+1\}}^{t} x^q_{ijs}u_{jt} + w_i \leq \sum_{t=0}^{T} R_{it}y^q_{it} + Cs^q_i, \quad q \in \bar{Q}^i,\ i \in I$$

$$u_{jt} \leq 0, \quad (j,t) \in B_1.$$

The solution $(\bar{u},\bar{w})$ to (4.9) defines the objective function of the column generation subproblem

$$\bar{c}^q = \min_{q\in Q} \left\{ \sum_{i\in I} \Big[- \sum_{(j,t)\in B_1} \sum_{s=\max\{0,t-p_i+1\}}^{t} \bar{u}_{jt}x^q_{ijs} + \sum_{t=0}^{T} R_{it}y^q_{it} + Cs^q_i - \bar{w}_i \Big] \right\}. \tag{4.10}$$

Now, since the set Q separates over I, subproblem (4.10) can be solved as $|I|$ independent subproblems (4.11),

$$\bar{c}^q_i = \min_{q\in Q^i} \left\{ - \sum_{(j,t)\in B_1} \sum_{s=\max\{0,t-p_i+1\}}^{t} \bar{u}_{jt}x^q_{ijs} + \sum_{t=0}^{T} R_{it}y^q_{it} + Cs^q_i \right\} - \bar{w}_i, \quad i \in I. \tag{4.11}$$

Each $\bar{c}^q_i < 0$ defines a new column (i,q) in the RMP as

$$\begin{bmatrix} Cs^q_i + \sum_{t=0}^{T} R_{it}y^q_{it} \\ \sum_{s=\max\{0,t-p_i+1\}}^{t} x^q_{ijs} \\ 1 \end{bmatrix}. \tag{4.12}$$

The decomposition of model (4.3) can be made analogously by separating constraints (4.3f) in the same way as (4.1f) above. A similar separation can be made for constraints (4.3g) by making use of (4.3h). Since $z_{jt} \leq b_{jt}$ for all $t \in \mathcal{T}$ and $j \in J_{\text{temp}}$, it holds in particular that $z_{jt} = 0$ for $(j,t) \in (J_{\text{temp}}, \mathcal{T})$ such that $b_{jt} = 0$. Therefore, constraints (4.3f) and (4.3g) are replaced by

$$x_{ijs} = 0, \quad s = \max\{0, t-p_i+1\}, \ldots, t,\ i \in I,\ (j,t) \in B_0, \tag{4.13}$$

$$\sum_{i\in I} \sum_{s=\max\{0,t-p_i+1\}}^{t} x_{ijs} \leq 1, \quad (j,t) \in B_1^{\text{perm}}, \tag{4.14}$$

$$\sum_{i\in I} \sum_{s=\max\{0,t-p_i+1\}}^{t} x_{ijs} \leq z_{jt}, \quad (j,t) \in B_1^{\text{temp}}, \tag{4.15}$$

where B_0, B_1 are defined as above and the latter has been separated into J_{perm} and J_{temp}. By letting constraints (4.14) and (4.15) remain in the MP and working through the above reformulations, the decomposition is achieved.

5
Numerical example

In this chapter, a numerical example of the in–house model in Chapter 4 is presented. The example is hypothetical and features six turbines of type Nordex N90 which have the PC specified in Table 5.1. In the implementation, the PC is specified for wind speeds from 0 to 20 m/s with 0.5 m/s increments and linear interpolation is used to define the power function $P(w)$ for all $w \in [0, 20]$. The type of maintenance required for each turbine is specified in Table 5.2, along with the expected maintenance duration.

w (m/s)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P (kW)	0	0	0	0	35	175	352	580	870	1237	1623	2043	2345	2475	2500

Table 5.1: *Default power curve of Nordex N90 wind turbines for a subset of wind speeds w. For wind speeds higher than 14 m/s, the power P is kept at 2500 kW.*

Turbine	Maintenance	p_i
1	PM	6
2	PM	2
3	PM	2
4	PM	4
5	PM	6
6	CM	4

Table 5.2: *Type of maintenance and duration of maintenance for each turbine $i = 1, \ldots, 6$ featured in the numerical example.*

The scheduling period of this example is from 2023-02-22–2023-02-24, meaning that $T = 72$. The wind speed forecast for this period, supplied by the company Rebase for use in this project, was issued at 12:00 on 2023-02-21 and has an average hourly absolute error of 1.7 m/s. The valid location of the forecast is at the coordinates of turbine 1 of Table 5.2. This means that all actual turbine data used in this example is related to turbine 1 and the rest of the turbines are thus considered as identical copies of the first. In any real application of the model, a seperate forecast would be needed at the location of each turbine considered. Figure 5.1 shows the wind speed forecast along with the hourly wind speed measured by turbine 1. The spot prices used are the actual spot prices during the scheduling period, i.e. not a forecast. The

results in this example are thus valid for the case when the error between forecasted prices and actual prices is zero. Furthermore, one maintenance team is working and the availability of the team, i.e. b_{jt}, is set to 07:00–16:00 for all three days.

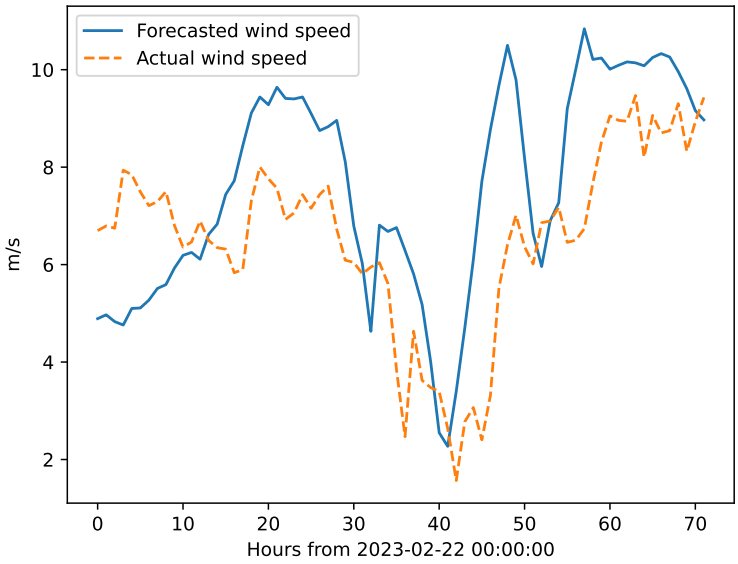

Figure 5.1: *Three-day wind speed forecast issued at 12:00 on 2023–02–21 and hourly wind speeds measured by turbine 1 in the numerical example.*

5.1 Results for the numerical example

The results of optimizing the maintenance schedule using the in–house model (4.1), for the example outlined above, can be seen in Figure 5.2. In this figure, the top graph shows the power forecast of one turbine during the scheduling period. This is obtained by—for each hour—inserting the corresponding wind speed forecast into the power function $P(w)$. The second graph shows the spot prices for each hour of the scheduling period while the third shows the revenue forecast obtained by multiplying the first two graphs for each hour. The final section of the figure shows the turbine downtime, i.e. the y-variable, for each turbine in the optimal solution found by Gurobi. In the case of CM, a red square specifies when maintenance is started on the corresponding turbine.

The problem is solved to optimality with all maintenance jobs being performed in the optimal solution and an optimal objective value of 2390. Using the optimal turbine downtime y^* together with the wind speed forecast $\{w_t\}_{t=0}^{T}$ and spot prices $\{e_t\}_{t=0}^{T}$, the expected lost revenue for turbine i is calculated as

$$R_i^{\text{forecast}} = \sum_{t=0}^{T} y_{it}^* P(w_t) e_t, \tag{5.1}$$

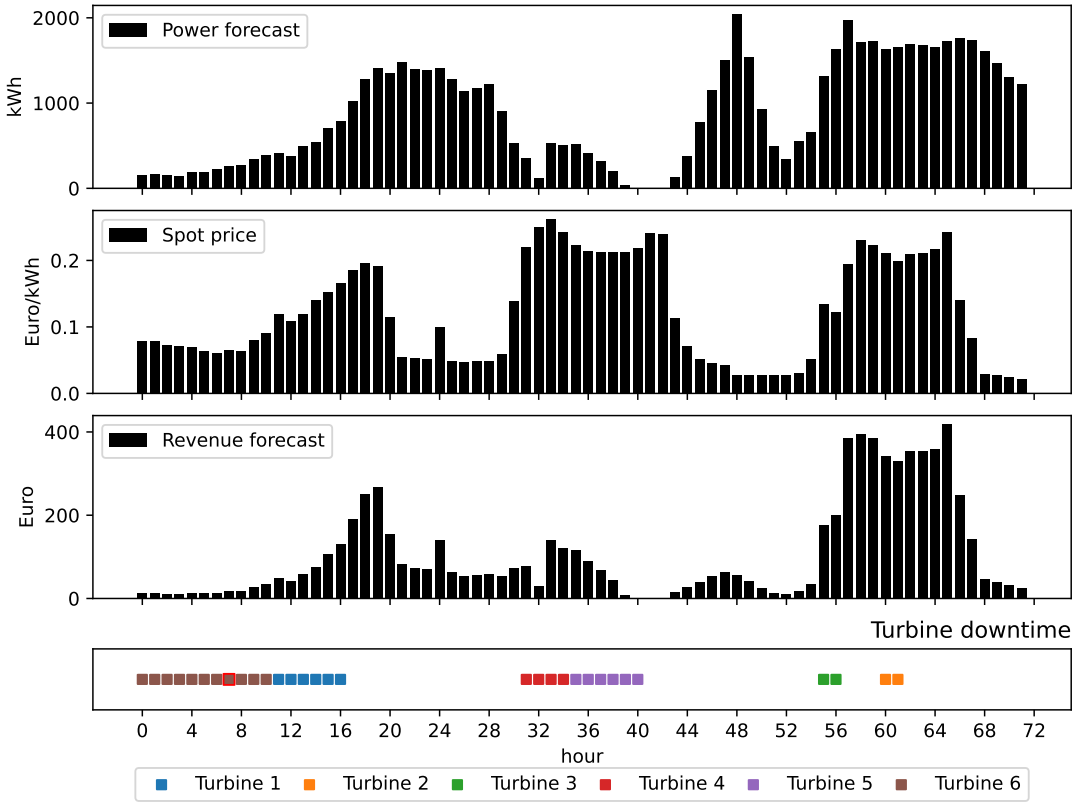

Figure 5.2: *Result of running model* (4.1) *for the example in Chapter 5 with one maintenance team. From the top, first: power forecast for the scheduling period, obtained using the power function* $P(w)$ *and wind speed forecast, second: spot prices for the scheduling period, third: revenue forecast obtained by multiplying the power forecast and spot prices for each hour, fourth: the turbine downtime in the optimal solution.*

the sum of which, in this case, coincides with the optimal objective value 2390 euro since $s_i^* = 0$ for all $i \in I$. By using the actual wind speed measurements shown in Figure 5.1 instead of the forecast, the same calculation yields $R_i^{\text{actual-wind}}$, i.e. the expected lost revenue for turbine i from the proposed maintenance schedule given the actual wind speeds. The difference $R_i^{\text{forecast}} - R_i^{\text{actual-wind}}$ represents the error in the expected revenue caused by non-perfect wind forecasts. There may still, however, be an error between the actual lost revenue and $R_i^{\text{actual wind}}$ caused by the turbines not performing according to the specified default PC. For this hypothetical example, in which turbine 1 was actually running during the entire scheduling period, the real hourly production P_t^{actual} is available for all turbines. Recall that all turbines are hypothetical copies of turbine 1 meaning they all had the same hourly production P_t^{actual} for $t = 0, \ldots, T$. The actual lost revenue can now be calculated as

$$R_i = \sum_{t=0}^{T} y_{it}^* P_t^{\text{actual}} e_t. \tag{5.2}$$

Note that this measure is not available when maintenance is performed but only in a historical analysis of the model. All the above results are summarised in Table 5.3,

which shows that the actual lost revenue was less than the expected lost revenue. This was expected given that the forecast overestimated wind speed, see Figure 5.1. It also shows that all turbines performed above expectation since $R_i > R_i^{\text{actual-wind}}$ holds for all $i \in I$.

Turbine i	Maintenance start, hour t	R_i^{forecast} (€)	$R_i^{\text{actual-wind}}$ (€)	R_i (€)
1	11	460	320	340
2	60	673	437	450
3	55	377	114	118
4	31	370	181	245
5	35	324	56	113
6	7	183	495	503
Total	–	2390	1607	1771

Table 5.3: *Summary of the lost revenues for the optimal solution to the numerical example with one maintenance team. The expected and actual lost revenues* R_i^{forecast}, $R_i^{\text{actual-wind}}$, *and* R_i *are calculated according to equations* (5.1) *and* (5.2)*, respectively*

By including another maintenance team, working the same hours as the first team, and running the optimization with otherwise unchanged input data, the results in the schedule in Figure 5.3 is found with the new optimal objective value of 1011. All these new results are summarised in Table 5.4. Note that the large drop in expected lost revenue is mainly attributed to no maintenance having to be scheduled during the third day when the revenue was expected to be large.

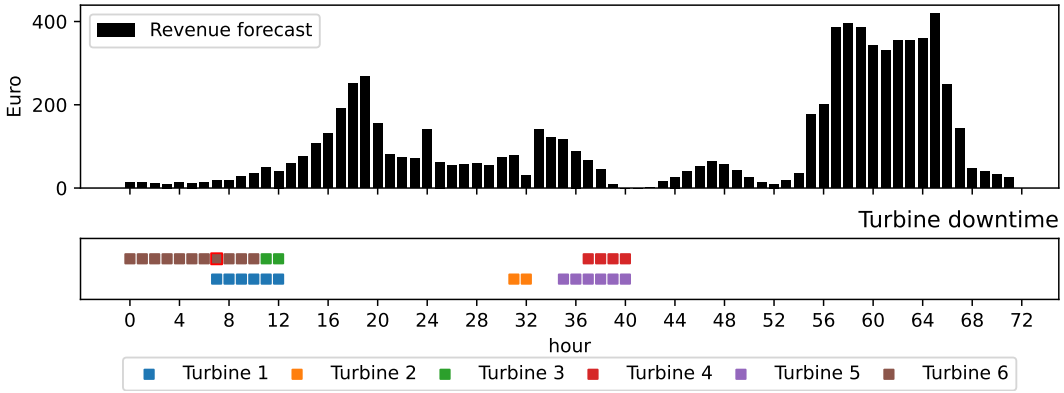

Figure 5.3: *Result of running the model* (4.1) *for the numerical example with two maintenance teams. Top: the same revenue forecast displayed in Figure 5.2. Bottom: turbine downtime in the optimal solution. See also caption of Figure 5.2.*

Turbine i	Maintenance start, hour t	R_i^{forecast} (€)	$R_i^{\text{actual-wind}}$ (€)	R_i (€)
1	7	187	263	277
2	31	107	62	105
3	11	89	116	120
4	37	118	50	89
5	35	324	56	113
6	7	183	495	503
Total	–	1011	1045	1210

Table 5.4: *Summary of the optimal solution to the numerical example with two maintenance teams. The actual and expected lost revenues* $R_i^{forecast}$, $R_i^{actual\ wind}$, *and* R_i *are calculated as in equations* (5.1) *and* (5.2).

6
Tests

In this chapter, the models are thoroughly tested using large data sets and randomly generated maintenance jobs. The errors in the wind speed forecast are analysed and the maintenance schedules are optimized in order to analyse the expected lost revenue. Finally, a run time analysis is performed. All computations in this chapter are performed on a laptop with a 2.3GHz AMD Ryzen processor with 8 available processors and 16 GB of RAM.

6.1 Forecast analysis

Three data sets have been used in the following tests. One set contains gross production and actual wind speeds of a reference turbine. The second contains actual hourly spot prices that have been collected using the Nord Pool API. As in Chapter 5, the results given in this chapter assumes the case of zero-error spot price forecasts. The final data set contains three–day hourly wind speed forecasts for the location of the reference turbine, issued at 12:00 every day from 2020-01-01 to 2023-03-31. Again as in Chapter 5, all turbines included in the tests are hypothetical copies of a reference turbine located at the valid coordinates of the forecast. This ensures that the available forecast is as good as possible. In a real application of the model, each turbine requires a specific forecast valid at its specific location.

By iterating over all forecast days, the hourly errors between actual wind speed and forecast can be calculated for each of the three days in a three–day forecast. Let

$$E_t^k = w_t^k - w_t^{\text{actual}}, \quad k = 1, 2, 3, \quad t = 0, \ldots, T-1, \tag{6.1}$$

where w_t^k is the forecasted wind speed for time step t issued k days earlier and w_t^{actual} is the actual, measured wind speed during time step t. For each $k = 1, 2, 3$, the mean μ_k and variance σ_k^2 of the errors can be calculated as

$$\mu_k := \frac{1}{T} \sum_{t=0}^{T-1} E_t^k; \tag{6.2}$$

$$\sigma_k^2 := \frac{1}{T} \sum_{t=0}^{T-1} (E_t^k - \mu_k)^2. \tag{6.3}$$

These expression are also known as the maximum likelihood estimators of the normal distribution. Figure 6.1 shows the distribution of errors, as defined in (6.1), for the three cases as histograms of the data and overlaid probability density functions of

normally distributed random variables with parameters given by equations (6.2) and (6.3). The graphs show that large errors are more common for forecasts further into the future. Another common measure of errors is the mean average percentage error

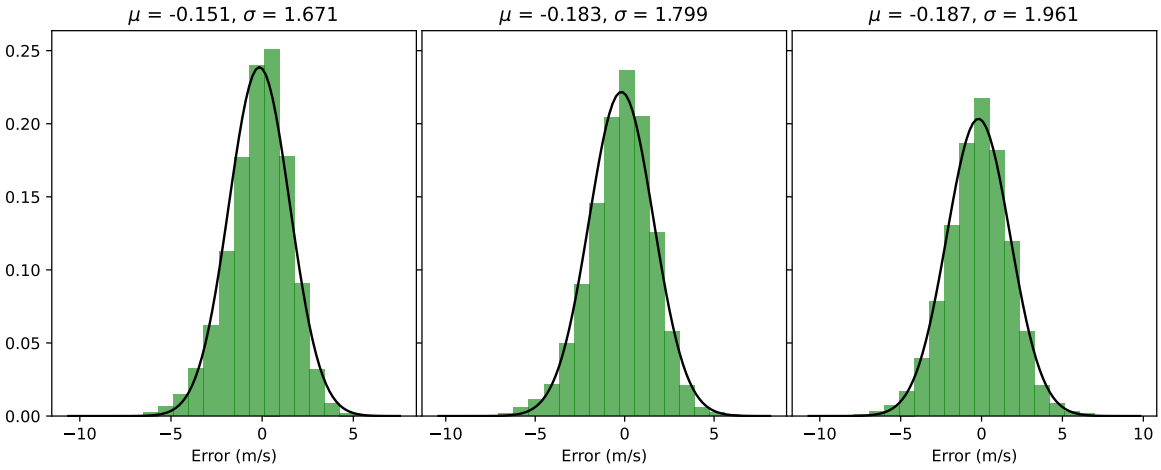

Figure 6.1: *Histograms of errors between forecasted and actual wind speed along with the probability density function of a normally distributed random variable with parameters calculated by* (6.2) *and* (6.3)*, respectively. From left to right: first, second, and third day of the three–day forecasts.*

(MAPE) defined for the kth day of the three–day forecast as

$$w^{\mathrm{MAPE},k} = \frac{1}{T}\sum_{t=0}^{T-1}\left|\frac{w_t^k - w_t^{\mathrm{actual}}}{w_t^{\mathrm{actual}}}\right|. \tag{6.4}$$

The value of $w^{\mathrm{MAPE},k}$ for $k = 1, 2, 3$ over the entire data set is 25.13%, 27.16%, and 29.97%, respectively.

6.2 Maintenance scheduling simulation

In order to analyse the maintenance scheduling models introduced in Chapter 4, hypothetical maintenance jobs need to be simulated. Given a maintenance team working from 07:00 to 16:00, as assumed in Chapter 5, it is reasonable to introduce two jobs per day, with maintenance durations between 1 and 6 hours, so that jobs do not accumulate over the course of the simulation. By iterating through the entire forecast data set in one day and three day increments, we simulate scheduling periods of one day and three days, respectively. Let p_i^{new}, $i = 1, \ldots, 2N$, $N \in \{1, 3\}$, be the expected duration of the new preventive maintenance jobs added for the current iterate, i.e. scheduling period. For each turbine i, let $p_i^{\mathrm{new}} \sim \mathrm{unif}\{1{,}6\}$ be a discrete, uniformly distributed random variable. In each iteration, the model (4.1) is solved with the scheduling horizon $T = 24N$ and with $p = (p^{\mathrm{old}}, p^{\mathrm{new}})$, where p^{old} is a list of durations of the uncompleted jobs from last iteration. Other parameters, such as the PC $P(w)$, is defined as in Chapter 5. After the problem has been solved to optimality the completed jobs are removed from p and $p^{\mathrm{old}} \leftarrow p$ for the next iterate.

Let R^{forecast} be the average daily expected lost revenue from a proposed schedule in the simulation. Furthermore, let $R^{\text{actual wind}}$ be the average daily expected lost revenue using actual wind speeds and let R be the average daily actual lost revenue using production data from the reference turbine. Letting M be the number of maintenance schedules in the simulation, these measures are calculated as follows

$$R_N^{\text{forecast}} = \frac{1}{N}\frac{1}{M}\sum_{m=1}^{M}\sum_{i\in I^m}\sum_{t=0}^{T} y_{it}^{(m)*} P(w_t^{\text{forecast}})e_t \tag{6.5}$$

$$R_N^{\text{actual wind}} = \frac{1}{N}\frac{1}{M}\sum_{m=1}^{M}\sum_{i\in I^m}\sum_{t=0}^{T} y_{it}^{(m)*} P(w_t^{\text{actual}})e_t \tag{6.6}$$

$$R_N = \frac{1}{N}\frac{1}{M}\sum_{m=1}^{M}\sum_{i\in I^m}\sum_{t=0}^{T} y_{it}^{(m)*} P_t^{\text{actual}} e_t, \tag{6.7}$$

where I^m is the set of maintenance jobs in iteration m and $y^{(m)*}$ are the optimal values of the y–variables in iteration m. The values of these measures after having run the simulation in one- and three–day increments with the number of permanent maintenance teams $|J| = 1$ and 2, respectively, are shown in Table 6.1. These results show that lost daily revenue, expected and actual, is lower if a three–day scheduling period is used. This is likely due to a greater flexibility in avoiding maintenance stops during days with high revenue forecasts.

$\lvert J\rvert$	$R_{N=1}^{\text{forecast}}$ €	$R_{N=1}^{\text{actual wind}}$ €	$R_{N=1}$ €	$R_{N=3}^{\text{forecast}}$ €	$R_{N=3}^{\text{actual wind}}$ €	$R_{N=3}$ €
1	230	232	208	177	199	181
2	177	199	181	99	149	140

Table 6.1: *Summary of the maintenance scheduling simulation, outlined in Chapter 6.2, with one–day and three–day scheduling periods and with one and two teams, respectively. The actual and expected daily lost revenues $R_N^{forecast}$, $R_N^{actual\ wind}$, and R_N are calculated as in* (6.5)*,* (6.6)*, and* (6.7)*, respectively*

The results presented in Table 6.1 also show that the reference turbine used in these tests in general performed below its default PC during the test period, since $R < R^{\text{actual wind}}$ in all cases. This is consistent with the performance data of the turbine. In these tests, the under performance of the turbine actually results in the expected lost revenue, R^{forecast}, sometimes being closer to the actual value than $R^{\text{actual wind}}$ is. This is still, however, to be considered as an error since, in practical use of the model, $R^{\text{actual wind}}$ is the measure of the actual lost revenue used in the absence of actual production, and should be as close to R as possible. A correction of this error can be achieved by using a best fit PC instead of the default PC used so far. Figure 6.2 shows the best-fit PC of the reference turbine using data on wind speeds and production from 2020-01-01–2023-03-31. Tests analogous to the ones presented above, using the same randomly generated maintenance jobs, now yield the results presented in Table 6.2. It is clear that using a best-fit PC reduces the

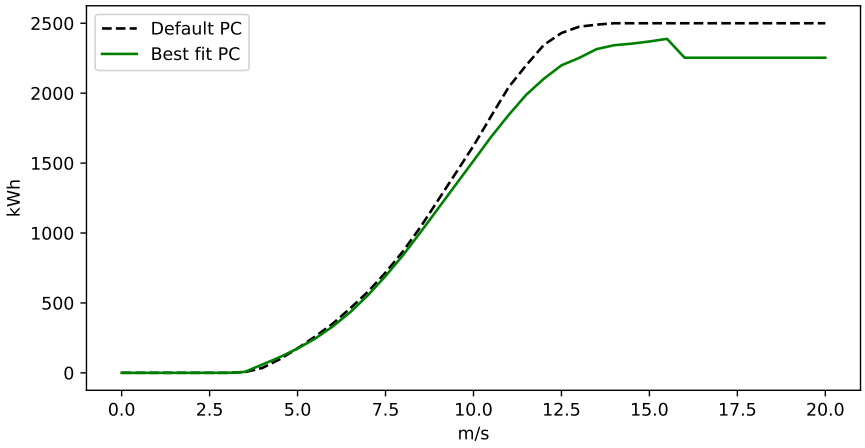

Figure 6.2: *The default PC and best fit PC for a wind turbine measured over the period 2020–01–01 to 2023–03–31*

$\|J\|$	$R^{\text{forecast}}_{N=1}$ €	$R^{\text{actual wind}}_{N=1}$ €	$R_{N=1}$ €	$R^{\text{forecast}}_{N=3}$ €	$R^{\text{actual wind}}_{N=3}$ €	$R_{N=3}$ €
1	220	221	207	171	191	181
2	186	201	189	97	144	140

Table 6.2: *Summary of the maintenance scheduling simulation using the best fit PC. The test performed is analogous to that in Table 6.1*

difference between $R^{\text{actual wind}}$ and R.

In order to investigate the benefit of including hourly spot prices in the optimization instead of, for instance, a constant average electricity price, the tests of Table 6.2 were run again, this time with a constant spot price. Choosing the price 0.07 euro/kWh, which is the average price over the entire simulation period, yields the actual lost revenues R^{const} presented in Table 6.3. Note that the constant prices are only used in the optimization, and not when calculating R^{const} during which the actual spot prices are used. These values show that optimizing the maintenance scheduling with constant prices results in higher actual lost revenues, i.e. a more expensive maintenance. Furthermore, by setting the constant spot price equal to zero, solving the model to optimality reduces to finding a feasible solution. Doing this, and calculating actual lost revenue R^{blind}, yields the last column of Table 6.3. This shows that the lost revenue can be reduced significantly by performing an optimization.

6.3 Run time analysis

In this chapter, a run time analysis is performed on the in–house model. The average computing time a number of specified problems using different solvers and relaxations are measured. There are nine problems specified by the number of maintenance jobs $|I| \in \{2, 4, 6\}$ and number of permanent teams $|J| \in \{1, 2, 3\}$. For all

$\|J\|$	$R_{N=1}$ (€)	$R^{\text{const}}_{N=1}$ (€)	$R^{\text{blind}}_{N=1}$ (€)	$R_{N=3}$ (€)	$R^{\text{const}}_{N=3}$ (€)	$R^{\text{blind}}_{N=3}$ (€)
1	207	232	253	181	203	234
2	189	232	258	140	171	235

Table 6.3: *The result of running the simulation presented in Table 6.2 with constant spot prices. R^{const}_N denotes the actual lost revenue with respect to actual spot prices, when a constant price has been used in the optimization and an N-day scheduling horizon has been used. R^{blind}_N denotes actual lost revenue when schedules are simply feasible solutions to the model. The values for R_N are copies from Table 6.2, for reference.*

cases, the scheduling horizon T is 72 hours, the maintenance duration p_i of all i is 4 hours, and the scheduled working hours for the teams are 07–16, as in the previous examples and tests. These problems are solved in four different ways. The first uses the solver glpk with binary requirements on all variables. The second also uses glpk but with binary requirements only on the x-variables according to the method for faster run times described in Chapter 4.3. The third way uses the Dantzig–Wolfe decomposition, also described in Chapter 4.3, with glpk as the solver of the D–RMP (4.9) and subproblems (4.11). The implementation of this method is described in Algorithm 1; note specifically the choice of starting columns in which $s_i = 1$ for all $i \in I$ and all other variables equal 0. These columns produce the worst possible objective value but are nonetheless feasible. The final way uses the Gurobi solver with binary requirements on x.

The results are shown in Table 6.4 where several entries are listed as > 3600. For these cases, no solution was consistently found within an hour after which the run was terminated. The run time results in Table 6.4 show that the commercial

|I|	|J|	glpk [s]	rel. glpk [s]	D–W [s]	Gurobi [s]
2	1	0.3	0.2	5.2	0.3
4	1	>3600	0.9	18.8	0.5
6	1	>3600	83.0	37.1	0.6
2	2	0.7	0.4	7.7	0.5
4	2	>3600	82.6	24.6	0.7
6	2	>3600	450.2	64.7	0.9
2	3	2.1	0.7	8.1	0.5
4	3	>3600	>3600	27.5	0.9
6	3	>3600	>3600	73.9	1.3

Table 6.4: *Average CPU computing times in seconds for a number of cases specified by the number of maintenance jobs and teams using different methods of optimization.*

state–of–the–art solver Gurobi solves all problems the fastest. The more interesting comparisons are between the other methods. It is clear that the LP relaxation of all variables except x allowed for more complex problems to be solved in reasonable time. Furthermore, the column generation through a Dantzig–Wolfe decomposition

Algorithm 1 Dantzig–Wolfe decomposition

$x^1_{ijt} \leftarrow 0$, $y^1_{it} \leftarrow 0$, $s^1_i \leftarrow 1 \quad \forall\, i \in I,\ j \in J,\ t \in \mathcal{T}$ ▷ Initial column
Include x^1, y^1, s^1, $\quad q \leftarrow 1$
while any $\bar{c}^q_i < 0$ **do** ▷ Column generation
    Build the RMP using x^q, y^q, s^q generated so far
    Solve the D–RMP $\Rightarrow$ dual variables u, w
    for $i \in I$ **do**
        Solve subproblem with integer req. on $x_i \Rightarrow \bar{c}^q_i, x_i, y_i, s_i$
        if $\bar{c}^q_i < 0$ **then** ▷ Negative reduced cost
            $q \leftarrow q + 1$
            $x^q_i, y^q_i, s^q_i \leftarrow x_i, y_i, s_i$
            Include x^q_i, y^q_i, s^q_i
        end if
    end for
end while
Solve RMP $\Rightarrow x^*, y^*, s^*$ ▷ Solve for final solution

effectively reduced run time for a number of cases and allowed problems to be solved in cases when the relaxed glpk method failed to do so in any reasonable amount of time.

7
Conclusion

The in–house model formulated in this project successfully schedules maintenance jobs in wind farms so that the expected lost revenue is minimized. The average daily lost revenue was shown to be smaller with a three–day scheduling period than with a one–day period due to the flexibility of avoiding high–revenue days. The actual lost revenue was calculated using actual production data and it was shown that performing optimization, as opposed to scheduling blindly, greatly reduces cost of losing out on production during maintenance. Furthermore, it was shown that using actual hourly spot prices in the optimization, instead of employing a constant spot price, further reduced the lost revenue.

The input data to the model was analysed by measuring the error in the wind forecast and turbine performance. It was made clear that an accurate wind forecast is important for an accurate maintenance scheduling and, while not shown, the same can be expected from the spot price forecast. It was also shown that using a best–fit power curve, instead of the default manufacturer curve, leads to more accurate approximations of the actual lost production which is necessary for evaluating the model in action.

Finally, a computation time analysis was performed for different scheduling cases. The methods of reducing the computing time presented in Chapter 4.3 were shown to be effective in allowing optimal scheduling of more complex instances in reasonable time. The commercial solver Gurobi was, however, superior in terms of speed.

8
Further work

The work presented in this thesis can be continued in numerous ways. First the models themselves can be extended with more constraints to allow for new functionality such as priority of maintenance jobs. The models can also be reworked to better describe all maintenance costs by, for instance, adding a transportation cost to the objective function and adding the necessary constraints. This may become more relevant when larger systems of turbines are considered.

Second, more work can be done to ensure that the revenue forecast is as accurate as possible. An idea brought forth by the company Rebase is to forecast power output from a wind park for each maintenance schedule considered. This would be beneficial since each combination of turbine downtime implies a specific wind speed loss on the other turbines and optimizing with this information included may lead to lower maintenance costs. Furthermore, tests need to be made using spot price forecasts when these are made available by Rebase.

Finally, more work can be done on the implementation of the Dantzig–Wolfe decomposition to achieve even shorter computing times. This includes improvements to the code and possibly a complete implementation of the Branch and Price algorithm discussed in Chapter 3.2.2.

Bibliography

[1] Nord pool. https://www.nordpoolgroup.com/en/. Accessed: 2023-06-21.

[2] Prognosuppföljning. https://www.smhi.se/data/meteorologi/prognosuppfoljning. Accessed: 2023-02-10.

[3] Rebase. https://www.rebase.energy/. Accessed: 2023-06-21.

[4] Francois Besnard, Michael Patriksson, Ann-Brith Strömberg, Adam Wojciechowski, and Lina Bertling. An optimization framework for opportunistic maintenance of offshore wind power system. In *2009 IEEE Bucharest PowerTech*. IEEE, June 2009.

[5] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. Springer International Publishing, 2014.

[6] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[7] Gregor Giebel and George Kariniotakis. Wind power forecasting—a review of the state of the art. In *Renewable Energy Forecasting*, pages 59–109. Elsevier, 2017.

[8] Tarjei Kristiansen. A time series spot price forecast model for the Nord Pool market. *International Journal of Electrical Power & Energy Systems*, 61:20–26, October 2014.

[9] Jan Lundgren, Mikael Rönnqvist, and Peter Värbrand. *Optimization*. Studentlitteratur, 2010.

[10] Jakub Nowotarski and Rafał Weron. Recent advances in electricity price forecasting: A review of probabilistic forecasting. *Renewable and Sustainable Energy Reviews*, 81:1548–1568, January 2018.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY