Solving a real-life, large-scale energy management problem

Godskesen, Steffen Elberg; Jensen, Thomas Sejr; Kjeldsen, Niels; Larsen, Rune

Published in:
Journal of Scheduling

Link to article, DOI:
10.1007/s10951-012-0279-8

Publication date:
2013

Document Version
Early version, also known as pre-print

Citation (APA):
Godskesen, S. E., Jensen, T. S., Kjeldsen, N., & Larsen, R. (2013). Solving a real-life, large-scale energy management problem. *Journal of Scheduling*, *16*(6), 567-583. https://doi.org/10.1007/s10951-012-0279-8

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

Solving a real-life, large-scale energy management problem

Steffen Godskesen · Thomas Sejr Jensen · Niels Kjeldsen · Rune Larsen

Received: date / Accepted: date

Abstract This paper introduces a three-phase hybrid heuristic for a large-scale energy management and maintenance scheduling problem. The problem is to schedule maintenance periods and refueling amounts for nuclear power plants with a time horizon of up to five years, and handling a number of scenarios for future demand and prices. The goal is to minimize the expected total production cost. The first phase of the heuristic solves a constraint programming model of a simplified version of the problem, the second performs a local search, and the third handles overproduction in a greedy fashion.

This work was initiated in the context of the ROADEF/-EURO Challenge 2010. In the concluding phase of the competition, our team ranked second in the junior category and sixth overall.

After correcting a small implementation bug in the program that was submitted for final evaluation, our solver ranks first in the overall results from the competition.

Keywords maintenance scheduling · constraint programming · ROADEF/EURO Challenge 2010 · production planning · hybrid heuristics

1 Introduction

In France, the majority of the electricity is produced by thermal — and in particular nuclear — power plants, and the main supplier is Électricité de France. There are two types of thermal power plants in the portfolio: *type 1 plants* which can be supplied with fuel continuously and without interrupting the production, and *type 2 plants* (nuclear power plants) which must be taken offline for refueling at regular intervals. The process of taking a power plant offline for maintenance is called an *outage* and the period between two outages a *production campaign*. While type 1 plants are more flexible than type 2 plants, the production cost incurred per unit of electricity is larger than for type 2 plants. Outages for type 2 plants should be scheduled such that the demand for electricity is satisfied at the lowest possible cost. A schedule for outages must satisfy a large number of constraints due to for example limited resources and safety considerations. Some of the constraints are due to limits on fuel levels in connection with each outage: There is a maximal fuel level for a power plant, a maximal allowed fuel level when a plant is taken offline, and a minimum refueling amount at each outage. An outage lasts a couple of weeks and is always scheduled to begin at the start of a week.

When planning future production, one must decide i) the timing of each outage, ii) refuel amounts, and iii) electricity production levels. The total production of electricity is not allowed to exceed the demand, and therefore type 2 plants can sometimes produce at less than their maximum production level. This situation is called *modulation*, and due to technical reasons there is a limit on the allowed modulation for each plant for each production campaign. By modulating nuclear power plants can adjust to small variations in electricity demand. Time is discretised into *time steps* spanning a couple of hours to model the variations in demand.

When a type 2 plant is taken offline for refueling and maintenance, the electricity must be produced by alternative sources. This can either be done by one of the other type 2 plants or by the more expensive type 1 power plants. The future electricity demand and the price of fuel for produc-

Steffen Godskesen · Thomas Sejr Jensen · Rune Larsen
Dept. of Mathematics and Computer Science
University of Southern Denmark
Campusvej 55, 5230 Odense M, Denmark

Niels Kjeldsen (✉)
DONG Energy A/S
Kraftværksvej 53, 7000 Fredericia, Denmark
Tel.: +45-99559667
E-mail: nielk@dongenergy.dk

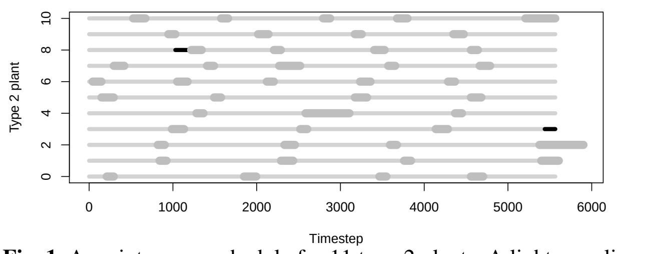

Fig. 1 A maintenance schedule for 11 type 2 plants. A light gray line is used for time steps with production, a wide dark gray segment for outages, and a black segment for when a plant is out of fuel.

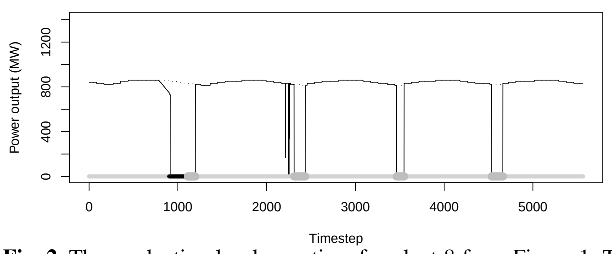

Fig. 2 The production level over time for plant 8 from Figure 1. The full line is the current production level and the dashed line the maximum allowed production level.

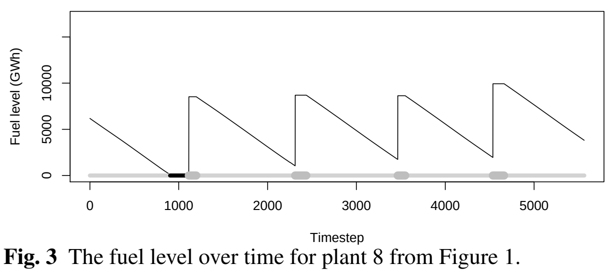

Fig. 3 The fuel level over time for plant 8 from Figure 1.

tion on type 1 plants is uncertain. This uncertainty should be taken into account in the planning of outages for type 2 plants. This is modeled by minimizing the *expectation* of future production cost over a number of given scenarios. The decisions on outage placement and refueling are common across all scenarios, whereas the decisions on production can be adjusted for individual scenarios.

1.1 The ROADEF/EURO Challenge 2010

The described problem was the subject of the ROADEF/-EURO Challenge 2010, which ran from July 2009 through June 2010. The competition was jointly organized by the French Operational Research and Decision Support Society, the European Operational Research Society, and the European utility company Électricité de France. In total 44 teams from 25 countries signed up for the challenge, of these 21 qualified for the final round, and 16 submitted a program for the final evaluation. The submitted programs were evaluated on ten problem instances, five known in advance and five only used in the evaluation. For each instance the time limit imposed on the program was one hour. In the concluding phase of the competition our team ranked second in the junior category and sixth overall. A complete description of the competition and evaluation rules can be found in Porcheron et al (2010).

The problem instances from the challenge have up to 75 type 2 plants, up to 120 scenarios for future prices and demand, and up to 5817 discrete time steps. Outages always start at the beginning of a week, and since the time horizon is about five years, there can be up to 277 possible outage start dates. This leads to a large-scale energy management problem.

A solution to the problem can be divided into two parts: a *maintenance schedule* and a *production plan*. A maintenance schedule specifies when outages of type 2 plants are scheduled and the amount of fuel to reload at each outage. A production plan specifies the production level of each plant for every time step and every scenario. To give an example of what a maintenance schedule might look like, see Figure 1, which shows a schedule for 11 type 2 plants.

Figures 2 and 3 gives an example of the production level and fuel level over time for plant number 8. Note the sudden decrease in production around time step 2200; this is modulation to ensure that there is no overproduction in the specific time step.

The following factors indicate that the problem is computationally hard to solve. First of all, it is NP-complete in the strong sense, as we prove in Section 2.5. Furthermore, the problem instances are large: a single problem instance takes up to 262 megabytes of hard disk space and contains more than $50 \cdot 10^6$ continuous decision variables just to represent production levels for every plant, time step, and scenario. Finally, there is a large number of constraints on production levels, fuel levels, refueling amounts, and scheduling of outages.

The program we submitted contained a small implementation bug which resulted in infeasible solutions for two of the five unknown instances (feasible solutions were found for all other instances). After correcting the program, we are able to find feasible solutions for all instances, and the solutions are actually better than those of the winning team.

1.2 Related work

A problem similar to the one studied here has been considered in Fourcade et al (1997). They consider roughly the same scheduling problem as here and formulate a mixed integer programming model. In their model, there is no decision variable concerning refueling amounts; this decision is instead handled as a predefined fixed amount. There is also no uncertainty of future demand and prices, and the demand is given per week, in contrast to the competition where the electricity demand is given in time steps of several hours, i.e., their discretisation is more coarse-grained.

They are able to solve small problems of up to 20 nuclear power plants with a MIP solver. The authors also attempted to tackle a model with 40 power plants, which yields a feasible solution after more than eight hours of computational time but with a significant gap to the LP lower bound. They report that this model is almost the size of the complete model of France which has 54 nuclear plants. The performance of MIP solvers and hardware have increased significantly since 1997, so modeling that particular problem as a mixed integer programming problem might be feasible with state of the art solvers.

Besides the work by Fourcade et al. very little is published specifically on nuclear power maintenance scheduling with refueling. The problem is mentioned by Dunning et al (2001), but the objective considered is to minimize the environmental impact and the problem only considers a single plant.

Several other power plant maintenance scheduling problems have been studied in the literature, see for example Satoh and Nara (1991); Yellen et al (1992); Kim et al (1997); Marwali and Shahidehpour (1999); Burke and Smith (2000). They all consider maintenance scheduling without refueling constraints, a time discretization of one week and future uncertainty is handled with a deterministic value for demand combined with an imposed reliability constraint. Several methods have attempted to optimize schedules, among these are: MIP, Benders decomposition, dynamic programming, genetic algorithms, simulated annealing, tabu search and combinations of these methods. Some of the work have scheduling constraints very similar to ones considered here, see Mukerji et al (1991); Silva et al (1995). The previous work in the literature does not consider power production and future uncertainty in the same detail as required in the competition setting.

The problem we consider includes complicating refueling constraints, many scenarios modeling future uncertainty and a detailed production planning part. So the work done in the context of the ROADEF Challenge 2010 is on a more general problem than the previous published work on scheduling power plant maintenance. It is possible to adapt the work done here to the settings above.

Setting production levels for power plants is treated in the literature under the term ‘economic dispatch’ — i.e., the problem of dispatching units to produce power in an economic way. While many settings have been considered, see for example Chowdhury and Rahman (2002), there are new features in the production planning for nuclear power plants. These features concern special bounds on production levels when the fuel level is low, which lead to nonlinear constraints. When the fuel level of a type 2 plant drops below a given threshold, a predetermined decreasing production profile is imposed. Without this constraint the production planning could be solved as a linear programming problem.

The scheduling part of this problem is similar to the Resource Constrained Project Scheduling Problem (RCPSP), see e.g. Neumann et al (2003). In both problems, activities (in this case outages) have to be scheduled subject to temporal constraints and limited resources. However, the problem at hand includes several constraints not found in common variants of RCPSP, such as disjunctive temporal constraints that only apply if a pair of activities is scheduled in a specified interval (as described in further detail in Section 2.3).

In the AIMMS Optimization Modeling Competition from 2009, organized in connection with the MOPTA conference, a truck maintenance scheduling problem was considered with some similarities to the nuclear power maintenance scheduling problem. There are to our knowledge no published work, except for a number of reports by the participating teams on the AIMMS website[1].

1.3 Our contributions

In this paper we prove that the considered energy management problem is NP complete in the strong sense, and propose a three phase hybrid optimization approach to obtain high quality solutions to the problem. Hybrid methods are popular in recent literature, see Blum et al (2008); van Hentenryck and Milano (2011); Hooker (2011). The proposed method is a three-phase heuristic approach for the management problem introduced above. The key parts of our approach are an initial solution construction and a two-part solution improvement phase. A similar decomposition was also present in the best performing approach of the previous ROADEF competition, see Bisaillon et al (2009).

The proposed hybrid approach consists of a constraint programming (CP) model in which an initial solution to the complex scheduling problem is found by using approximated constraints for production levels and fuel consumption. From this first schedule we apply a stochastic local search (SLS) algorithm based on a simple neighborhood structure. Two essential components in the local search are a very fast feasibility check and a fast but approximated evaluation of the change in solution cost. To guide the search, we use a simulated annealing metaheuristic. In the third and final phase we use a greedy algorithm to remove any overproduction.

The approach is the result of an engineering cycle in which we recognized that the SLS had difficulties finding an initial feasible solution, while the task turned out to be easy when tackled by CP. On the other hand, the CP was not able

[1] http://www.aimms.com/community/modeling-competitions/mopta-2009

to improve the solution with respect to the objective function, something the SLS turned out to be more successful in doing.

Other elements that turned out to be important are the aggregation and approximation of the production planning in both the CP model and the local search.

1.4 Overview

The paper is organized as follows. Section 2 gives a formal description of the optimization problem as well as a complexity proof. In Section 3 we describe how to obtain a feasible solution using a CP model for scheduling outages and a heuristic for production planning. The stochastic local search phase is described in Section 4. Section 5 describes how overproduction is handled. Computational analysis and results are the topic of Section 6. Finally, we conclude in Section 7.

2 Problem description

Each type 2 plant goes through a number of *cycles*. A cycle is composed of an outage followed by a production campaign. During an outage the plant cannot produce electricity because of maintenance and reloading of fuel. During a production campaign the plant can produce electricity. Having type 2 plants producing at less than maximum capacity leads to wear of the equipment involved and should thus be avoided if possible. Modulation is the difference between maximum capacity and actual production.

The demand for electricity is not known with certainty at the time of planning. This uncertainty is modeled by introducing a number of *scenarios*, each of which represents a realistic future demand profile for the planning horizon discretised into a number of time steps. Optimizing for several realistic scenarios instead of just one generally leads to more robust plans.

Decisions concerning scheduling of outages and refueling amounts must be shared by all scenarios, but production levels can be determined for each individual scenario. This creates a dependency between scenarios, since the outage schedule and refueling amounts must be feasible with respect to every scenario's production plan.

The objective is to create both a maintenance plan and production plan that satisfy the demand for electricity at the lowest average cost over all scenarios. The cost must be minimized while satisfying a number of constraints. These constraints can be divided into four categories: i) bounds on production levels, ii) bounds on fuel levels and refueling amounts, iii) different kinds of temporal constraints on the scheduling of outages, and iv) bounds on the outages' simultaneous use of limited resources.

2.1 Decision variables and bounds

We use $s = 0,\dots,S-1$ to index scenarios, $t = 0,\dots,T-1$ to index time steps, $h = 0,\dots,H-1$ to index weeks, $j = 0,\dots,J-1$ to index type 1 plants, $i = 0,\dots,I-1$ to index type 2 plants, and $k = 0,\dots,K-1$ to index cycles. A week consists of a number of time steps, i.e., two different discretisations of the planning horizon are used. This is because outages are scheduled on a weekly basis, whereas a higher resolution is required for the productions levels. The length of a time step in hours is denoted by D (all time steps have the same length).

The length of outage k for type 2 plant i is denoted by $DA_{i,k}$. Let $ha(i,k) \in \mathbb{Z}$ denote the week that the k'th outage for type 2 plant i starts, and $TO_{i,k}$ and $TA_{i,k}$ denote the earliest and latest starting time for $ha(i,k)$, formally

$$TO_{i,k} \le ha(i,k) \le TA_{i,k}. \tag{1}$$

The bounds $TO_{i,k}$ and $TA_{i,k}$ may be undefined, in which case the corresponding inequality is trivially satisfied. If the upper bound is undefined for some outage, the outage does not have to be scheduled. If outage k for plant i is not scheduled then $ha(i,k) = -1$ and constraint (1) is not enforced. Outage $k+1$ for plant i cannot start before outage k for plant i is finished, and outage $k+1$ cannot be scheduled unless k is scheduled.

The amount of fuel reloaded at type 2 plant i in outage k is denoted by $r(i,k) \ge 0$ and if k is scheduled $r(i,k)$ must satisfy

$$RMIN_{i,k} \le r(i,k) \le RMAX_{i,k}, \tag{2}$$

where the bounds $RMIN_{i,k}$ and $RMAX_{i,k}$ are input data. If k is not scheduled, then $r(i,k) = 0$.

Let $p(\ell,t,s) \ge 0$ denote the production of plant ℓ (which may be of type 1 or 2) at time step t in scenario s.[2]

2.2 Auxiliary variables

In addition to the decision variables there is a number of auxiliary variables which can be derived from the decision variables and thus do not increase the size of the solution space.

The set of time steps composing the k'th outage of type 2 plant i is denoted by $ea(i,k)$, and the set of time steps composing the subsequent production campaign is denoted by $ec(i,k)$. For any outage k, the production $p(i,t,s)$ of plant i must be zero for every $t \in ea(i,k)$ and every scenario s.

The fuel stock of type 2 plant i at time step t in scenario s is denoted by $x(i,t,s) \ge 0$. The initial fuel level of plant i

[2] As in the original problem formulation citeproadefWeb we index decision variables using parentheses in order to distinguish them from parameters in the model.

(at time step 0) is denoted by XI_i and specified in the input data. During a production campaign for plant i the decrease in fuel level from time step t to $t+1$ in scenario s equals the production multiplied by the length of a time step D, and therefore

$$x(i,t+1,s) = x(i,t,s) - p(i,t,s)D. \tag{3}$$

During an outage the fuel level at type 2 plant i increases because of refueling. Due to technical reasons the new fuel level is not simply the sum of the old fuel level and the amount reloaded. Formally, if t is the first time step in outage k for plant i, then the new fuel level for i in scenario s is computed by

$$x(i,t+1,s) = Q_{i,k}x(i,t,s) + r(i,k) + Q'_{i,k}, \tag{4}$$

where $Q_{i,k} < 1$ and $Q'_{i,k}$ are input data[3].

2.3 Constraints

The problem constraints are here divided into four groups, namely production level constraints, fuel level constraints, scheduling constraints, and resource constraints.

Production level constraints Let $DEM^{t,s}$ denote the *demand* at time step t in scenario s. The total production must equal the demand in every scenario and every time step

$$\forall s,t : \sum_{j=0}^{J-1} p(j,t,s) + \sum_{i=0}^{I-1} p(i,t,s) = DEM^{t,s}. \tag{5}$$

Let $PMIN_j^{t,s}$ and $PMAX_j^{t,s}$ denote the minimum and maximum, respectively, *allowed production* of type 1 plant j at time step t in scenario s, then

$$\forall s,t,j : PMIN_j^{t,s} \leq p(j,t,s) \leq PMAX_j^{t,s}. \tag{6}$$

The bounds on production for a type 2 plant are more complex, since they depend on the current fuel stock of the plant. If the fuel level is above a specified threshold $BO_{i,k}$, then the production is bounded from above by $PMAX_i^t$

$$\begin{aligned}&\forall s,t,i,k : t \in ec(i,k) \wedge x(i,t,s) \geq BO_{i,k}\\ &\qquad \Rightarrow 0 \leq p(i,t,s) \leq PMAX_i^t.\end{aligned} \tag{7}$$

As long as the fuel level is above the threshold modulation is undesirable and therefore there is an upper bound $MMAX_{i,k}$ on the accumulated modulation of plant i in each production campaign k

$$\forall s,i,k : \sum_{\substack{t \in ec(i,k) \wedge \\ x(i,t,s) \geq BO_{i,k}}} (PMAX_i^t - p(i,t,s))D \leq MMAX_{i,k}. \tag{8}$$

[3] Equation (4) is a simplification of Equation (CT10) in the original model defined by ROADEF, but the two formulas are equivalent when appropriately adjusted values for $Q_{i,k}$ and $Q'_{i,k}$ are used.

If the fuel level is below the threshold $BO_{i,k}$, the upper bound decreases and a lower bound is also enforced. This is referred to as the *declining power profile*. How much the upper bound decreases for type 2 plant i in production campaign k is specified by the function $PB_{i,k}$ which maps fuel levels to real numbers between zero and one. Formally, for all s,t,i,k, if $t \in ec(i,k)$ and $x(i,t,s) < BO_{i,k}$, then the production must lie in a small interval centered around P_x

$$P_x = PB_{i,k}(x(i,t,s)) \cdot PMAX_i^t, \tag{9}$$

$$(1-\varepsilon)P_x \leq p(i,t,s) \leq (1+\varepsilon)P_x. \tag{10}$$

Note that if the plant produces at P_x and this implies that the plant runs out of fuel, it is not allowed to produce at all. Thus, (10) applies only if

$$x(i,t,s) \geq P_x \cdot D. \tag{11}$$

If (11) does not hold, $p(i,t,s)$ must be zero.

Fuel level constraints There are upper bounds on the *fuel level* before and after a type 2 plant's outage. Let $AMAX_{i,k}$ denote the upper bound on the fuel level at the time when outage k for plant i starts and $SMAX_{i,k}$ the upper bound on the fuel level after the outage k for plant i. If the k'th outage for plant i starts at time step t, inequality (12) and (13) must hold for every scenario s

$$x(i,t,s) \leq AMAX_{i,k}, \tag{12}$$

$$x(i,t+1,s) \leq SMAX_{i,k}. \tag{13}$$

Scheduling constraints There are disjunctive temporal constraints between outages. If a specified pair of outages (i,k) and (i',k') is scheduled such that both intersect a given interval (this interval may be the entire planning horizon), then the following constraint must be satisfied:

$$ha(i,k) - ha(i',k') \geq Se \vee ha(i',k') - ha(i,k) \geq Se', \tag{14}$$

where the lower bounds Se and Se' are input data.

Several types of temporal constraints are defined in the original problem definition from ROADEF citeproadefWeb, but they can all be converted to the type in (14).

Resource constraints For every week h there is a collection of subsets of outages. For each such subset A and an associated natural number N, at most N of the outages in A are allowed to be on outage in week h:

$$\forall h : \sum_{(i,k) \in A} \Phi(i,k,h) \leq N, \tag{15}$$

where $\Phi(i,k,h)$ equals 1 if plant i is in outage k in week h and 0 otherwise.

There are limited resources available for maintenance. To model this, a collection of subsets of outages is given.

Each subset B in the collection has an associated resource availability Q. For every B, at most Q of the outages in B can use maintenance resources in any week

$$\forall h: \sum_{(i,k)\in B} \Phi'(i,k,h) \leq Q, \tag{16}$$

where $\Phi'(i,k,h)$ equals 1 if outage (i,k) uses a maintenance resource in week h and 0 otherwise. Note that the weeks in which an outage uses resources are not necessarily the same as the weeks in which it is in outage. This difference is specified in the input data.

Finally, there is an upper bound on the capacity that is allowed to be offline at a given time in a given region. To model this, a collection of subsets of plants is given. Each subset C in this collection has an associated upper bound $IMAX$ and a subset of weeks IT. For every C, $IMAX$, and any week h in IT, the total offline capacity of plants in C is not allowed to exceed $IMAX$

$$\forall h \in IT : \forall t \in h : \sum_{\substack{i\in C:\exists k:\\ t\in ea(i,k)}} PMAX_i^t \leq IMAX. \tag{17}$$

Note that in (17) a week is considered as a set of time steps. The sum is simply over all type 2 plants in C that are offline at time step t.

2.4 Objective function

The objective function is composed of three terms: the total cost of refueling all type 2 plants, the average cost of type 1 production over all scenarios, and the value of residual fuel at type 2 plants at the end of the planning horizon. Let $C_{i,k}$ denote the cost of fuel for type 2 plant i in cycle k, $C_{j,t,s}$ the cost of production for type 1 plant j at time step t in scenario s, and C_i the value of fuel for type 2 plant i at the end of the planning horizon. Then the objective function to be minimized is

$$\sum_{i=0}^{I-1}\sum_{k=0}^{K-1} C_{i,k} r(i,k) + \frac{1}{S}\sum_{s=0}^{S-1}\sum_{t=0}^{T-1}\sum_{j=0}^{J-1} C_{j,t,s} p(j,t,s) D - \frac{1}{S}\sum_{s=0}^{S-1}\sum_{i=0}^{I-1} C_i \cdot x(i,T,s). \tag{18}$$

2.5 Computational complexity

To prove the NP-completeness of the problem under consideration, we propose a reduction from 1-in-3-SAT, which is proved to be NP-complete in the strong sense by Schaefer (1978). Reductions directly from a scheduling problem might be possible but is complicated by the exponential (albeit pseudo-polynomial) number of time steps that often will arise.

First note that a solution to the energy management problem can be checked in polynomial time, since it has a polynomial number of constraints and each of them can be checked in polynomial time.

Given a boolean formula $\beta_1 \wedge \cdots \wedge \beta_c$ where each clause β_i, $1 \leq i \leq c$, is the disjunction of three boolean literals from the set $\{x_1,\ldots,x_n\}$, 1-in-3-SAT asks for an assignment of true or false to $x_1,\ldots,x_n$ such that exactly one of the literals in each clause β_i evaluates to true.

To solve an instance of 1-in-3-SAT by using the problem under consideration, we construct an instance having a single scenario as follows. A type 2 plant i with a single outage with a duration of one week is created for each clause β_i. Furthermore, a week is created for each variable x_h and its negation $\neg x_h$, in such a way that the variable and its negation occupy successive weeks. Scheduling an outage in the week that corresponds to x_h (respectively $\neg x_h$) is interpreted as forcing x_h to be true (respectively false).

All outages must be scheduled in order for the 1-in-3-SAT instance to be satisfiable. A constraint of type (1) with bounds set to include all $2n$ weeks for every outage ensures this.

To ensure that the single outage for plant i can only be scheduled in one of the three weeks corresponding to literals in β_i, a constraint of type (17) which restricts the amount of offline capacity is added for the single outage. We set $PMAX_i^t = 1$ for all t and $IMAX = 0$. Furthermore, we let C contain plant i's single outage and let IT contain all weeks except the three corresponding to literals in the β_i. Constraint (17) is enforced on time steps rather than weeks, so the number of time steps is set to $2n$ such that there is one time step per week.

Constraints of type (14) are added to ensure that outages are not scheduled in both the week corresponding to x_h and $\neg x_h$. For each pair of clauses that contains literal x_h and $\neg x_h$ respectively, a constraint of type (14) is defined on the two weeks corresponding to x_h and $\neg x_h$ (which are consecutive by construction). Forcing a separation of two weeks between the corresponding plants' outages prevents these outages from being scheduled in the weeks corresponding to x_h and $\neg x_h$, respectively.

See Figure 4 for an example of a simple boolean formula encoded as a maintenance scheduling problem.

The construction described above is polynomial in the input size, as we have c clauses, giving rise to c outages, each with three valid weeks where it can be scheduled. For each of these weeks, we need less than c constraints to restrict it from conflicting with the other outages. The conversion is thus bounded from above by $3c^2$.

The remaining constraints are constructed such that they do not prevent scheduling of any type 2 plant in any week. A single type 1 plant can be used to cover any demand we decide on. We set the demand $DEM^{t,0} = I$ for each time step

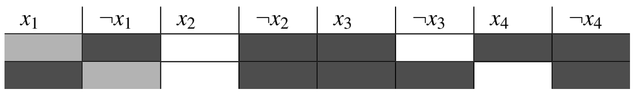

Fig. 4 A representation of the formula $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$. There is one row per clause in the formula. Dark gray weeks are disallowed using constraint (17). Outages can be scheduled in light gray and white weeks, but the light gray weeks for x_1 and $\neg x_1$ cannot both be used for outages due to a constraint of type (14).

t, and choose $PMAX_i^t = 1$, $PMIN_i^t = 0$ for each type 2 plant i and time step t. The initial fuel stock is set large enough to allow every type 2 plant to produce in all $2n$ weeks without outages. Minimum and maximum bounds on refueling of the type 2 plants are set such that they do not constrain the scheduling solution, i.e., $RMIN_{i,k} = 0$, $RMAX_{i,k} = 0$. To ensure that $AMAX_{i,k}$ and $SMAX_{i,k}$ do not become constraining, they are set to the initial fuel stock.

If and only if we can schedule all outages in the energy management problem, the 1-in-3 SAT instance is satisfiable. Truth values are assigned to literals in the given 1-in-3-SAT instance as follows. A literal is set to true if some outage is scheduled in the corresponding week and to false otherwise. Thus, any instance of 1-in-3-SAT can be solved by scheduling outages. As mentioned above, the size of the reduction's output is polynomial in the size of the 1-in-3-SAT instance and can obviously be constructed in polynomial time, and therefore the optimization problem in this paper is NP-complete in the strong sense, i.e., it is NP-complete even if all numerical parameters are encoded in unary base.

2.6 Hybrid approach overview

Our solver constructs a solution in three phases.

1. In the first phase, solving a CP model creates an initial maintenance schedule with decisions on starting week and refueling amount for each of the scheduled outages. The CP model does not include any decisions on production levels for type 1 or type 2 plants.
2. In the SLS phase of the approach, the maintenance schedule is improved (both the placement of outages and the refueling amounts). The SLS uses the initial maintenance schedule but recalculates the refueling amounts. During the local search, production levels for the type 2 plants are the same for all scenarios. This is done to speed up evaluation of the quality of the current maintenance schedule. The evaluation is an approximation since calculating the true cost for all scenarios is computationally too expensive to be used in the local search procedure.
3. In the final phase, production levels for both type 1 and 2 plants for every scenario are calculated. In this phase the production levels of type 2 plants are also adjusted to remove overproduction if any is present.

Note that a full solution with production levels for every scenario only is available at the very end of the solution process.

3 Initial solution construction

We showed that the maintenance scheduling part of the problem is NP-complete, and our experience is that finding non-trivial feasible solutions to this problem is challenging in practice as well. Our initial approach included different directions: a set of construction heuristics combined with SLS and a CP approach. Only the CP approach consistently gave feasible maintenance schedules. Our overall setup thus starts by creating a first feasible maintenance schedule using CP.

3.1 Constraint programming

An exact representation of the problem would result in a very large number of variables, since it requires modeling of every time step. To reduce the size of the model, we aggregate all data indexed by time steps into weekly equivalents. Furthermore, concepts such as modulation, the decreasing power profile, and the cost of type 1 production lead to an excessively large model. Therefore, we focus primarily on finding a feasible maintenance schedule and introduce a surrogate objective function that approximates the actual objective function (18), thereby leaving the rest of the optimization to the subsequent SLS.

The surrogate objective function guides the solver towards a solution with an appropriate number of outages. Too few outages will make the type 2 plants run out of fuel and thus be unproductive for an extended period of time. Too many outages take the plant offline unnecessarily and might cause infeasibility, as the plants are unable to use enough fuel for the next outage to take place.

A CP model is used to find a feasible maintenance schedule.[4] For every outage (i,k), the CP model has three decision variables: A binary variable $\sigma(i,k)$ deciding if outage k for type 2 plant i is scheduled or not, the integer variable $ha(i,k)$ determining the starting week for the outage, and the integer variable $r(i,k)$ determining the refueling level. Refueling amounts are continuous in the problem formulation but are discretised because most CP solvers cannot handle continuous variables. To reduce the domain of the refuel variables in the CP model, the discretisation is into segments of 1000 fuel units.

We model the scheduling and resource constraints (14)-(17) exactly but approximate the fuel level constraints because an exact representation requires exact modeling of

[4] For a general introduction to CP, see Apt (2003).

fuel consumption of type 2 plants, which would lead to a very large model, due to the many time steps.

The complete CP model is described in the subsections below. The sections describing the constraints and the objective function contains the technical details necessary for reimplementing the CP model, but readers not interested in doing so can skip ahead to Section 3.1.5.

3.1.1 Scheduling constraints

The following two sets of constraints enforce the time windows defined in (1) on outages. For every outage (i,k) with lower bound $TO_{i,k}$ defined

$$\sigma(i,k) = 1 \Rightarrow ha(i,k) \geq TO_{i,k}. \tag{19}$$

For every outage (i,k) with upper bound $TA_{i,k}$ defined

$$\sigma(i,k) = 1 \Rightarrow ha(i,k) \leq TA_{i,k}. \tag{20}$$

For every outage (i,k) with upper bound $TA_{i,k}$ defined, we enforce the following constraint since (i,k) must be scheduled

$$\sigma(i,k) = 1. \tag{21}$$

To ensure that outages are scheduled in order, we enforce the following constraint for each type 2 plant i and every outage $k = 1,\ldots,K-1$

$$ha(i,k-1) + DA_{i,k-1} \leq ha(i,k), \tag{22}$$

where $DA_{i,k}$ is the length of outage (i,k). Since scheduling of an outage requires scheduling of all previous outages for the same type 2 plant, we enforce the following constraint for each type 2 plant i and every outage $k = 1,\ldots,K-1$

$$\sigma(i,k) \leq \sigma(i,k-1). \tag{23}$$

To enforce the temporal constraints in (14), we add the following constraint for every given pair of outages (i,k) and (i',k'), interval of weeks $[L,U]$, and lower separation bounds Se and Se' in weeks

$$\begin{aligned}&\sigma(i,k) = 1 \wedge \sigma(i',k') = 1 \wedge\\ &L < ha(i,k) + DA_{i,k} \wedge ha(i,k) \leq U \wedge\\ &L < ha(i',k') + DA_{i',k'} \wedge ha(i',k') \leq U \Rightarrow\\ &ha(i,k) - ha(i',k') \geq Se \vee ha(i',k') - ha(i,k) \geq Se'.\end{aligned} \tag{24}$$

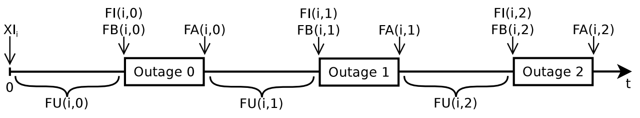

Fig. 5 Temporal overview of notation for approximated fuel usage (FU) and fuel levels (XI, FI, FB, FA) for a type 2 plant.

3.1.2 Resource constraints

To ensure that at most a specified number of type 2 plants are offline in a week h, we implement constraints as in (15) with $\Phi(i,k,h) = 1$ if

$$\sigma(i,k) = 1 \wedge ha(i,k) \leq h \wedge h \leq ha(i,k) + DA_{i,k}, \tag{25}$$

and $\Phi(i,k,h) = 0$ otherwise.

To ensure that enough resources are available in every week, we implement constraints as in (16) with $\Phi'(i,k,h) = 1$ if

$$\sigma(i,k) = 1 \wedge ha(i,k) + LU_{i,k} \leq h < ha(i,k) + LU_{i,k} + TU_{i,k}, \tag{26}$$

and $\Phi'(i,k,h) = 0$ otherwise. Here, since the weeks in which an outage (i,k) uses resources are not necessarily identical to the weeks in which it is offline, $LU_{i,k}$ and $TU_{i,k}$ specify the resource usage's offset and duration, respectively.

To ensure that there is sufficient type 2 capacity online, we implement constraints based on (17) as follows

$$\forall h \in IT : \sum_{i \in C} \sum_{k=0}^{K-1} \Phi(i,k,h) \sum_{t \in h} PMAX_i^t < IMAX, \tag{27}$$

where $\Phi(i,k,h)$, as above, is defined by (25) and IT, C, and $IMAX$ as in (17).

3.1.3 Fuel level approximation

To enforce the bounds on refuel amounts, we implement the following constraints based on (2)

$$r(i,k) \geq \sigma(i,k) RMIN_{i,k}, \tag{28}$$

$$r(i,k) \leq \sigma(i,k) RMAX_{i,k}. \tag{29}$$

To estimate the fuel level before and after each outage, we need to introduce several variables, which are visualized in Figure 5.

Recall that XI_i is the initial fuel level at plant i. Let $\beta(i,h)$ denote the accumulated fuel usage for type 2 plant i in weeks 0 through $h-1$ of the planning horizon, assuming production at maximum capacity

$$\beta(i,h) = D \sum_{h'=0}^{h-1} \sum_{t \in h'} PMAX_i^t. \tag{30}$$

The $\beta(i,h)$ values are used to estimate the fuel usage during the production period preceding outage k for type 2 plant i which we denote by $FU(i,k)$. For $k=0$ we have

$$FU(i,0)=\sigma(i,0)\beta(i,ha(i,0)), \tag{31}$$

and for $k>0$ we have

$$FU(i,k)=\\ \sigma(i,k)(\beta(i,ha(i,k))-\beta(i,ha(i,k-1)+DA_{i(k-1)})). \tag{32}$$

The definitions of the following three variables are mutually recursive, since they depend on values for the previous outage. An initial estimate $FI(i,k)$ of the fuel level at type 2 plant i at the time of outage k is obtained by subtracting the estimated production from the fuel level at the previous outage:

$$FI(i,k)=\begin{cases} XI_i-FU(i,k), & \text{if } k=0 \\ FA(i,k-1)-FU(i,k), & \text{if } k>0, \end{cases} \tag{33}$$

where $FA(i,k-1)$ denotes the estimated fuel level after outage $k-1$ for type 2 plant i and is computed by (34) which is derived from (4)

$$FA(i,k)=Q_{i,k}FB(i,k)+r(i,k)+Q'_{i,k}. \tag{34}$$

Finally, we adjust the initial estimate $FI(i,k)$ for the fuel level before an outage. The problem with $FI(i,k)$ is that the declining power profile in constraint (10) is ignored. This will often underestimate the actual fuel level because the plant usually produces at less than $PMAX$ at the end of the production campaign and consequently uses less fuel. Experiments show that this often leads to situations where no feasible production plan can be found for a given schedule. Thus, in order to take the declining power profile into account, we adjust $FI(i,k)$ if it is low enough that the power profile is activated in the end of the production campaign. More precisely, if $FI(i,k)<BO_{i,k}$, we assume that plant i would have run out of fuel when $FI(i,k)=-BO_{i,k}$ and make a linear interpolation. Note that the decision to choose $-BO_{i,k}$ as the cut-off point is heuristic. The adjusted fuel level estimate $FB(i,k)$ is computed by

$$FB(i,k)=\begin{cases} 0, & \text{if } FI(i,k)\le -BO_{i,k} \\ FI(i,k), & \text{if } FI(i,k)\ge BO_{i,k} \\ FI(i,k)+ & \\ \frac{1}{2}(BO_{i,k}-FI(i,k)), & \text{otherwise.} \end{cases} \tag{35}$$

This relationship between $FB(i,k)$ and $FI(i,k)$ is shown in Figure 6. This approximation implies that a feasible solution might be infeasible in the CP model and vice versa, but in practice the approximated constraints leads to maintenance schedules that are feasible in the sense that there exists a feasible production plan for all scenarios.

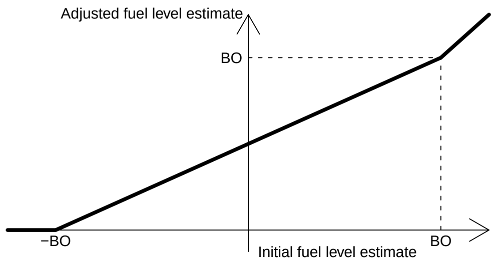

Fig. 6 The piecewise linear function mapping an initial fuel level estimate $FI(i,k)$ to an adjusted estimate $FB(i,k)$ that takes the power profile into account. The adjusted estimate is never negative, and for any $FI(i,k)\ge BO$, the function works like the identity function.

All the described constraints are implemented as standard relational constraints.

3.1.4 Surrogate objective function

The scheduling problem is primarily concerned with minimizing the use of type 1 plants, which is equivalent to maximizing the amount of online type 2 capacity. Therefore, the surrogate objective function measures average offline type 2 capacity over time. Minimization of this objective function ensures good decisions on the number of outages scheduled. Note that if too few outages are scheduled the plants run out of fuel, hence we also include this effect in the objective function.

Let α_i denote the average maximal fuel usage per week for type 2 plant i

$$\alpha_i=\frac{\sum_{t=0}^{T-1}PMAX_i^t}{H}D. \tag{36}$$

Furthermore, let the auxiliary decision variable k'_i denote the index of the last scheduled outage for type 2 plant i, i.e.

$$k'_i=\sum_{k=0}^{K-1}\sigma(i,k)-1. \tag{37}$$

Then, the surrogate objective function to be minimized is

$$\sum_{i=0}^{I-1}\alpha_i\Bigg(\Big[ha(i,0)-\frac{XI_i}{\alpha_i}\Big]^+ +\sum_{k=1}^{K-1}\Big[\sigma(i,k)\\ \Big(ha(i,k)-ha(i,k-1)-DA_{i(k-1)}-\frac{FA(i,k-1)}{\alpha_i}\Big)^+\Big]\\ +\Big[H-ha(i,k'_i)-DA_{ik'_i}-\frac{FA(i,k'_i)}{\alpha_i}\Big]^+\Bigg). \tag{38}$$

The first term is the estimated offline capacity before each plant's first outage, the second term is estimated offline capacity before each of the plants' subsequent outages, and the third term is the estimated offline capacity after each plant's last scheduled outage.

3.1.5 Search strategy

A CP solver finds a solution to an optimization problem by searching a tree which is pruned by applying *constraint propagation* and *branch and bound* strategies. Two decisions are crucial for making this pruning effective, namely *variable selection* (choosing the next variable to branch on) and *value selection* (choosing a value for the chosen variable). Variable and value selection strategies are generally chosen according to the *fail-fast* principle, which says that if no feasible solution exists in the current sub tree, then the search should determine this as early as possible. Furthermore, finding a good solution early in the search is desirable because it improves the efficiency of branch and bound pruning.

Our variable selection strategy attempts to make consecutive decisions in the search tree that affect outages which are likely to be related by constraints; this often corresponds to outages that are scheduled close to each other. This is achieved by branching on variables grouped by outage number in the following way. A random permutation π of type 2 plants is constructed. Let $\pi(\ell)$ denote the ℓth plant in this permutation. We examine all cycles $k \in \{0,\dots,K-1\}$, and for each k examine all type 2 plants $\pi(\ell) \in \{0,\dots,I-1\}$. For each outage $(\pi(\ell),k)$ the variables are fixed in the following order and with the specified value selection strategy:

1. Determine whether outage $(\pi(\ell),k)$ is scheduled or not, i.e., whether $\sigma(\pi(\ell),k) = 1$ or $\sigma(\pi(\ell),k) = 0$. The $\sigma(\pi(\ell),k) = 1$ branch is considered first, since this leads to more scheduled outages.
2. In the $\sigma(\pi(\ell),k) = 1$ branch, determine the outage starting week $ha(\pi(\ell),k)$. The earliest possible week is considered first, since this leaves more room for subsequent outages for plant $\pi(\ell)$.
3. Determine the refuel amount $r(\pi(\ell),k)$. The maximal amount is considered first, since this leads to more type 2 capacity.

Preliminary experiments showed that this branching strategy works well. The CP solver is given at least 10 minutes to find a solution, if a solution is found the search is stopped, and if not it is continued until a feasible solution is found. Details about the CP solver software are given in Section 6.2, and details about the time allocation between CP and SLS are given in Section 6.3.

3.2 Greedy production level planning

We set the production levels $p(i,t,s)$ and refueling amounts $r(i,k)$ of a feasible maintenance schedule returned by the CP solver by means of a greedy algorithm which we call *production planner*. Note that the refueling amounts from the CP maintenance schedule are not used in the phases after CP.

Only constraint (5) concerning demand binds production across different type 2 plants. The production planner ignores the demand and therefore all computations can be done for each type 2 plant independently. Ignoring the demand may lead to overproduction which is fixed by modulation in a final phase, described in Section 5.

The algorithm starts with the first time step and goes through all time steps in the planning horizon. It uses the initial fuel level to produce at maximum capacity until no more fuel remains or the next outage is encountered. If a plant runs out of fuel in some production campaign, it cannot produce in the rest of the production campaign.

We use the production planner in two settings:

1. It is used on an initial maintenance schedule from the CP solver, and in this case the production planner initially sets refuel amounts to the minimum allowed amount $RMIN_{i,k}$ for every outage.
2. It is used repeatedly in the SLS, where the current refuel amounts are updated. The production planner first tries to achieve feasibility by reducing refuel amounts, as described in Section 3.2.1, and then to increase the refueling amounts, as described in Section 3.2.2.

3.2.1 Reducing refuel amounts

Infeasibility with respect to fuel levels can occur if constraint (12) is violated because of a too high fuel level before an outage, or constraint (13) is violated because of a too high fuel level after an outage. If one of these situations is encountered, the production planner backtracks to the previous outage and reduces the amount of refueling done there — this change is subject to constraint (2) on refueling amounts. This is done recursively, and if the backtracking reaches the start of the planning horizon without resolving the problem, the planner declares the maintenance schedule infeasible. If this happens and the production planner is called from the local search, the infeasible neighbor is simply skipped. It has never happened to the initial solution from the CP solver, but in such a case the problem could be returned to the CP to attempt to create a new solution.

3.2.2 Increasing refuel amounts

After having decreased refueling amounts wherever necessary to the point where constraints (12) and (13) on fuel levels are satisfied, the production planner tries to increase the refuel amounts as much as possible in order to maximize type 2 production. This is done for each production campaign k in turn, starting with the campaign after the last scheduled outage and proceeding backwards. If plant i is able to produce at $PMAX_i^t$ for all $t \in ec(i,k')$ where $k' \geq k$, then there is no need to increase refuel amounts (unless the value of type 2 fuel at the end of the planning horizon exceeds its cost at the time of outage k; we ignore this possibility). Otherwise, the production planner repeatedly tries to increase the refueling amount by $0.02(RMAX_{i,k} - r(i,k))$ until either production is maximized or the maximal allowed refuel amount is reached.

4 Improvement by stochastic local search

When evaluating the quality of the initial maintenance schedule from the CP we see that it is of relatively low quality. This is the case since the strategy used in the CP model is to place outages mainly to ensure a feasible solution and to a lesser extent to minimize production costs. This leaves room for improving the temporal placement of outages.

Moving outages can reduce the total cost of production in two ways. First, it can increase the amount of electricity produced by type 2 plants and thereby decrease the production of the more expensive type 1 plants. Second, it can move outages to a time period where the type 1 production costs are lower.

The SLS is done in a typical simulated annealing setting, see Kirkpatrick et al (1983). The basic move in the local search is to choose a random outage and shift it a few weeks forward or backward. After each move the scheduling and resource constraints (1) and (14) to (17) are checked for feasibility. If the constraints are satisfied, the production planner calculates updated refueling amounts and production levels in order to check feasibility with respect to fuel levels. If a move is feasible, it is candidate for being accepted by the simulated annealing depending on the current temperature and the change in the objective function.

A limitation of the SLS procedure is that the number of outages is not changed. The effect of this is that the CP model is trusted with deciding the number of outages for each power plant.

We have tried to let the local search add and remove outages but found that this operation introduces efficiency problems. Adding or removing outages leads to changes in the whole solution, meaning that the local search spends a lot of time re-adjusting the placement of all the other outages. This gives a very long evaluation time (several hundred iterations) for adding or removing an outage, and thus hinders an effective local search.

In the next sections, we describe the SLS approach in detail.

4.1 Neighborhood

Formally, given a schedule for outages, $ha(i,k)$, $0 \leq i < I, 0 \leq k < K$, a neighboring scheduling solution ha' is obtained by applying the move (i',k',m)

$$ha'(i,k) = \begin{cases} m, & \text{if } i = i' \text{ and } k = k' \\ ha(i,k), & \text{otherwise.} \end{cases} \tag{39}$$

The value m is chosen uniformly random in the interval $\left[TO_{i',k'}, TA_{i',k'}\right]$, so only neighboring schedules that satisfy constraint (1) are considered. A move (i',k',m) thus corresponds to selecting outage k' of plant i' and moving it to start in week m.

The size of the neighborhood is bounded from above by $I \cdot K \cdot H$, but the bounds in (1) reduce the number of neighbors significantly. The length of the interval $[TO_{i,k}, TA_{i,k}]$ is usually between 20 and 30 weeks on average (including outages where $TO_{i,k}$ or $TA_{i,k}$ are undefined, in which case the interval consists of all weeks before $TA_{i,k}$ or all weeks after $TO_{i,k}$). In two instances where very few outages have constraints of type (1), the average interval is around 150 weeks. The size of the neighborhood is reduced by only considering moves (i,k,m) where the outage is moved less than n weeks forward or backward, i.e.,

$$\left|ha_{i,k} - m\right| < n. \tag{40}$$

Preliminary experiments have shown that a good value for n is 20, this is used for all instances.

The feasibility of a neighbor can be checked effectively because each outage is involved in a relatively low number of constraints. It is straightforward to precompute a data structure that maps an outage to a list of the scheduling constraints involving the outage. With this list, feasibility of a neighbor can be checked efficiently. If the feasibility check detects a violated constraint, the evaluation is terminated immediately and the move is discarded. This implies that the SLS never moves to an infeasible maintenance schedule.

4.2 Delta evaluation and acceptance criteria

From the previous section we have that scheduling feasibility can be checked efficiently. Calculating the change in solution cost and checking for refueling feasibility is however more involved. For a type 2 power plant that had an outage moved it is necessary to re-plan the production for all time steps. Only a single artificial scenario of production levels for each type 2 plant is maintained, meaning that we only recalculate production levels and implied fuel levels for one scenario. This is done both to ensure that we still have a feasible production plan and to estimate the change in the objective function. The new production plan is calculated by the production planner described in Section 3.2. If the production planner is unable to find a feasible production plan, the move is discarded.

The change in objective consists of three parts: the change in cost of type 2 refueling Δ_{refuel}, the change in cost of type 1 production Δ_{type1}, and the change in value of remaining fuel at the end of the planning horizon $\Delta_{remainder}$. The total change Δ is then

$$\Delta = \Delta_{refuel} + \Delta_{type1} - \Delta_{remainder}. \tag{41}$$

We now describe how to compute each of these three contributions.

When the greedy production planner has calculated new production levels for the single scenario and updated the refueling amounts, we get

$$\Delta_{refuel} = \sum_{k=0}^{K-1} C_{i,k}(r'(i,k) - r(i,k)), \tag{42}$$

where $r'(i,k)$ is the refuel amount in the neighboring solution and $C_{i,k}$ the fuel cost in outage (i,k).

The amount of fuel remaining at the end of the planning horizon can vary from scenario to scenario. Here we estimate the remaining amount assuming that every type 2 plant has full production in all time steps, meaning that the estimate is a lower bound on the actual amount of remaining fuel. This gives

$$\Delta_{remainder} = C_i(x'(i,T,s^*) - x(i,T,s^*)), \tag{43}$$

where $x'(i,T,s)$ is the fuel level in the neighboring solution, C_i the value of remaining fuel, and s^* is the artificial full production scenario.

Calculating Δ_{type1}, the change in total cost of type 1 production, is more complicated and described in Section 4.3.

When the objective function change Δ for a feasible move has been calculated, it is considered by the simulated annealing. A neighbor is accepted with probability

$$p = \min\left(1, \exp\left(-\frac{\Delta}{\tau}\right)\right), \tag{44}$$

where τ is the current temperature.

The initial temperature is dynamically set such that approximately half of the considered moves are accepted at the beginning of the search. The cooling scheme is geometric with cooling ratio c and is applied every $n_{plateau}$ iterations. When m_{idle} consecutive iterations have been considered without any move being accepted, a restart is performed. When the search is restarted, the current temperature is set to $k_{restart}$ times the starting temperature of the previous annealing run.

At the end of the time allocated to the SLS, a first-fit descent is performed to ensure that the search reaches a local optimum.

4.3 Estimation of type 1 production cost

First we explain how the exact change in type 1 production cost is calculated. Then we introduce an effective way to approximate this change, since the exact computation is too expensive to be suited for use in local search.

To help calculating Δ_{type1} efficiently, we maintain a list of the total type 2 production in every time step. We look at the type 2 plant which had an outage moved and use the change in production in each time step, to update the list of total type 2 production. In scenarios where the demand is higher than the total type 2 production, the difference must be covered by type 1 plants, which have a fixed cost per unit of power produced.

The type 1 plants can be preordered by increasing production costs, and thus the exact change in total type 1 cost, given a change in total type 2 production, can be computed in $O(ln(J))$ time for a single time step and scenario, as explained below.

Assume that the previous total type 2 production in time step t was $p_{T2}(t)$ and after a move to a neighboring maintenance schedule, the total type 2 production is adjusted to $p'_{T2}(t)$. To calculate the cost of the residual type 1 production for a scenario s we examine the ordered list of type one power plants. While the residual demand $DEM^{t,s} - p'_{T2}(t) - p'_{T1}(t)$, where $p'_{T1}(t)$ is the current type 1 production, is positive we increase the production of the type 1 plant with the lowest production cost and available production capacity. At some point the residual demand will be satisfied by the extra type 1 production.

The above procedure can be seen as a piecewise linear function for the cost in each scenario, mapping type 2 production to the cost of covering the residual demand using type 1 plants. Finding the linear piece of this function that corresponds to a given total type 2 production can be done by binary search, which gives a computational complexity of finding the total type 1 production cost for a single time step of $O(ln(J))$.

The above procedure gives the type 1 production cost for a single time step and a single scenario. Summing this exact change over all time steps and scenarios gives a total complexity of $O(T \cdot S \cdot ln(J))$ which is too slow for local search.

Hence we approximate the change in cost. The approximation removes the need to consider all scenarios and the binary search through the set of type 1 plants but not the need to consider all time steps, making the complexity of the approximation $O(T)$.

To estimate the change in type 1 cost for a single time step, we precompute a single piecewise linear function which maps total type 2 production to the (exact) total type 1 cost for each time step. This is done by summing all S piecewise linear functions for a single time step.

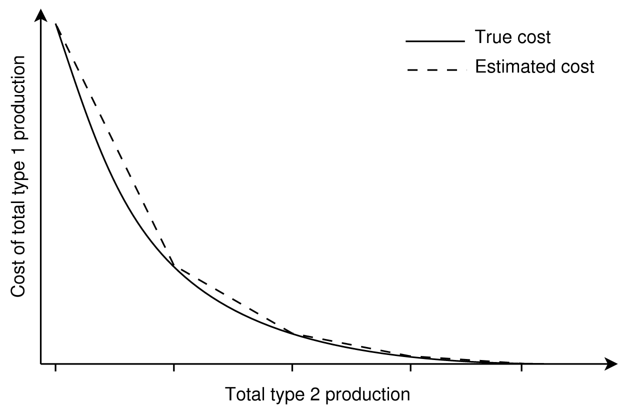

Fig. 7 Linear approximation of the total type 1 costs for a single time step.

The solid line in Figure 7 shows an example of such a function. The curve seems smooth but is, since it is a sum of piecewise linear functions, also a piecewise linear function. As the total type 2 production increases, the need for type 1 production diminishes, and when total type 2 production reaches the maximum demand over all scenarios, the total type 1 cost is zero.

The many breakpoints make the evaluation of the function computationally expensive because it must go through the breakpoints in order to find the interval containing the current total type 2 production (a binary search can speed up this process but is still too slow). Therefore, we approximate it with another precomputed piecewise linear function, which has fewer breakpoints chosen such that they are equidistant on the x-axis. The dashed line in Figure 7 shows an example of this approximation. Since the actual type 1 cost is convex, the approximation is an upper bound on the actual cost.

Experiments show that this approximation is relatively good, even for a reasonably small number of breakpoints. We have chosen to use $3 \cdot I$ breakpoints, and the approximation error has been analyzed on all instances. If absolute type 1 production demands under 100 units are ignored, the worst average deviation over an instance is $\approx 3.36 \cdot 10^{-4}$ percent per measurement. This is significantly less than the differences in cost encountered during local search. When evaluating the very small type 1 production demands we ignored above, the relative approximation error is significant, but this does not hinder the SLS, since these moves only give correspondingly small changes in the objective value.

Using the precomputed approximation we can perform a constant time evaluation of the total type 1 cost for a given total type 2 production in a single time step. No search for the stored breakpoint closest to the given total type 2 production is required, because the breakpoints are equidistant. If the total type 2 production falls between two stored breakpoints, we use linear interpolation to get the total type 1 cost. If the total type 2 production in time step t is P_2^{total}, the interpolation for the type 1 production cost is done between the indices:

$$i_{low} = \left\lfloor \frac{P_2^{total}}{Int} \right\rfloor, \qquad i_{high} = \left\lceil \frac{P_2^{total}}{Int} \right\rceil, \tag{45}$$

where P_2^{total} is the sum of type 2 production in time step t, and Int is the length of the interval between two breakpoints.

The approximation of the change in total type 1 cost is very fast compared to the exact computation but slow when compared to the scheduling constraint violation check, since the approximation requires a computation for every time step. Note however that since we only move to feasible solutions, the approximation is only performed if all the scheduling constraints are satisfied.

5 Modulation

In this last phase of the solution process we calculate production levels for all scenarios and ensure that there is no overproduction in any scenario. This gives the final solution to the energy management problem. In this phase, we do not move any outages but do adjust refueling amounts for some of the outages.

From the SLS we have a maintenance schedule where the cost of covering demand by type 1 power plants is minimized. This may lead to solutions with overproduction at type 2 plants due to the variations in demand across scenarios. Formally, we have in some time step t in some scenario s

$$\sum_{i=0}^{I-1} p(i,t,s) > DEM^{t,s}.$$

Such overproduction can be eliminated in two ways: modulation can be used to decrease the power output of a type 2 plant in the affected time step, or refueling can be decreased such that a type 2 plant runs out of fuel before that time step. Making a power plant run out of fuel is not an attractive option because it eliminates the rest of its production for the current campaign across all scenarios, since refueling amounts are shared among all scenarios. It will also limit the plant's fuel level in the next production campaign. Therefore this method is used as a last resort, i.e., modulation is used whenever possible — and it is capable of eliminating overproduction in all the examined instances.

There are three constraints on the amount of modulation that can be performed on a single power plant: constraint (8) restricts the amount of modulation that can be performed in the current campaign, and constraints (12) and (13) enforce an upper limit on the fuel level before and after refueling. The latter must be handled by adjusting the refueling of the power plant, but this will affect all scenarios.

Since refueling amounts for a plant are shared among all scenarios, we use a two-step procedure to eliminate overproduction. First, we ensure that there exists a refueling scheme that is feasible for all scenarios using the minimum demand scenario defined below. Second, we try to adjust modulation for each individual scenario to reduce the need for expensive type 1 production.

We define a minimum demand scenario to be $DEM^t_{min} = \min_s(DEM^{t,s})$ for all t. By using this scenario we can ensure feasibility of all scenarios, since the only way the demand influences feasibility is by making modulation necessary to get the type 2 production low enough to match demand in constraint (5). As type 2 power plants according to Porcheron et al (2010) deliver 87% of the combined power on average, such situations arise sparingly, and the minimum demand scenario can be modulated to feasibility in all instances used in the competition.

5.1 Modulation and refueling for the minimum demand scenario

A modulation and refueling scheme is created for the minimum demand scenario, thus ensuring that the refueling is feasible for all scenarios. This is done by examining the time steps in increasing order. When a time step with overproduction is detected, a target plant i is selected among the type 2 plants. Plant i has its production lowered until there is no more overproduction or as much as feasible while respecting the remaining modulation capacity in that campaign, see equation (8). This reduces the fuel consumption, meaning that the fuel level constraints must be checked. If necessary, we repair the refueling amounts at plant i using the greedy production planner. If this fails, the modulation on plant i is undone, and another plant is selected. We continue selecting a new plant and reducing its production until we have eliminated any overproduction in the current time step. Note that since we consider the minimum demand scenario, we are guaranteed that there is no overproduction in any scenario.

Since the cost of modulating each type 2 plant is the same, modulation can be seen as a resource that expires when the production campaign ends. Thus, the target plant selection strategy iterates through plants in non-descending order of their current production campaign's end date.

A refuel plan may be infeasible for the minimum demand scenario but still be feasible for all scenarios. Our method is unable to cope with this situation and will therefore declare such a schedule infeasible, but this has never occurred in any instance used in the competition.

5.2 Modulation per scenario

Having determined modulation for the minimum demand scenario, we fix refuel amounts and apply the modulation algorithm from the previous subsection on each scenario. Modulating without consecutive adjustment of the refueling amounts can yield infeasible fuels levels, meaning that modulation cannot always be done for a plant. In this case the algorithm moves on to the next type 2 plant.

On some instances this step improves the objective value by more than 1%. This improvement was done after the competition.

6 Computational analysis and results

In this section we describe the problem instances used for computational tests, implementation details, how much time is spent in different components of the heuristic, tuning of the parameters in the simulated annealing algorithm, and finally we report the numerical results.

6.1 Problem instances

We have tested our algorithm on ten real-life instances supplied by Électricité de France. Table 1 shows various figures as well as the best known objective value for each of the instances, which are taken from the ROADEF website citeproadefWeb. Note that the B and X instances are pairwise very similar, e.g. $B6$ and $X11$, $B7$ and $X12$ and so on. This similarity is due to the fact that they are based on the same data, but have different demand profiles.

Instance	File size	T	Weeks	S	J	I	Best solution
B6	140	5 817	277	50	25	50	83 424 716 217
B7	144	5 565	265	50	27	48	81 174 243 138
B8	262	5 817	277	121	19	56	81 926 206 073
B9	262	5 817	277	121	19	56	81 750 858 197
B10	252	5 565	265	121	19	56	77 767 024 999
X11	140	5 817	277	50	25	50	79 116 772 289
X12	143	5 523	263	50	27	48	77 589 910 940
X13	262	5 817	277	121	19	56	76 449 207 715
X14	262	5 817	277	121	19	56	76 172 998 633
X15	250	5 523	263	121	19	56	75 101 398 439

Table 1 Overview of the ten instances showing for each instance file size in megabytes, number of time steps (T), number of weeks, number of scenarios (S), number of type 1 plants (J), number of type 2 plants (I), and the objective value of the best known solution.

6.2 Implementation details

The algorithms are implemented in Java, and the scheduling problem is solved using the Gecode CP solver version 3.3.1 citepgecode. The version of our program that was submitted for the qualifying phase used ILOG's CP Optimizer version 2.3 instead of Gecode, but the former model was unable to solve the larger problem instances used in the final round. Moreover, the Gecode solver allows the user control over the applied branching strategy. On the basis of preliminary observations, we decided to stop the CP solver after ten minutes and then return the best solution found. If no feasible solution has been found after ten minutes, we let the solver run until the first feasible solution is found.

The strategy described in Section 3.1 finds a feasible schedule solution in less than two minutes for all instances, as seen in Table 2. The table also shows the minimum time needed to construct a solution, using the initial maintenance schedule from the CP and applying the modulation phase but without SLS.

	B6	B7	B8	B9	B10
CP (s)	84	67	11	22	28
Mod (s)	7	7	48	43	43
Total (s)	167	153	256	237	263
% gap	12.17	6.73	16.26	19.79	9.80

	X11	X12	X13	X14	X15
CP (s)	13	8	13	9	15
Mod (s)	8	10	70	71	49
Total (s)	108	94	315	366	242
% gap	9.09	5.80	14.97	15.52	7.37

Table 2 The time in seconds needed to produce the first feasible scheduling solution (CP), modulating it (Mod) and total time including reading from and writing to disk (Total). The last row is the percent-wise gap from best known objective value.

In the competition a machine with a 2.50Ghz Intel Xeon processor was used, while our corrected results were created using a 2.13GHz Intel Xeon processor. In both cases 8GB of main memory were available.

6.3 Time allocation

Preliminary tests indicated that ten minutes for the CP solver and the remaining 50 minutes for local search and other tasks is a reasonable distribution of the one hour available. The tests showed that the CP solver finds a first solution very fast, but by letting the solver continue the search, solutions with more scheduled outages for each type 2 plant are found. This is important since the subsequent local search does not change the number of scheduled outages. However almost all improvement from scheduling more outages appear within the first ten minutes the CP solver runs, and the search is therefore stopped after ten minutes.

In cases where more than one hour of computation time is available, it could be interesting to provide the SLS with several solutions from the CP, each with a different number of outages. Since the convergence of the SLS takes a significant amount of the allocated 50 minutes this was not possible in the context of the competition.

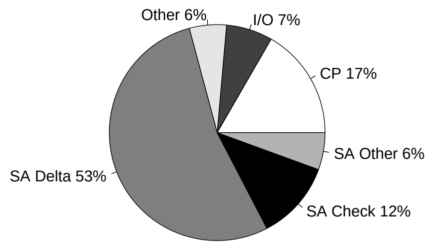

Fig. 8 How one hour of wall clock time is spent when solving instance B10.

The pie chart in Figure 8 shows how much time is usually spent in different parts of the program. Not surprisingly, most time is spent on the simulated annealing algorithm, whose delta evaluation alone accounts for more than half of the total running time. This is due to the fact that evaluation of a neighbor requires replanning of production levels which is very time consuming, even when applying the approximation described in Section 4.2. Somewhat unusual is the 7% of the total running time spent on reading an instance and writing a solution to the hard disk, which is caused by the very large instance and solution files. The latter takes up to 950 megabytes of hard disk space.

6.4 Tuning the stochastic local search

To determine a set of parameters that perform well for all problem instances we compared a number of different parameter settings. All the following combinations of parameters were run for all ten instances in sets B and X and for ten different random seeds. In order to reduce the number of configurations, we decided after preliminary testing to fix the number of moves at each temperature to 100 and that the initial temperature after a restart should be twice the initial temperature of the previous run.

Cooling ratio $c = \{0.95, 0.96, 0.97, 0.975, 0.98, 0.985, 0.99, 0.995\}$
Start acceptance ratio = $\{0.25, 0.5, 0.75\}$
Stopping criteria $m_{idle} = \{25, 50, 75, 100, 125, 150, 175, 200, 300, 500, 800\}$
Number of moves per temperature plateau $n_{plateau} = 100$
Reheat constant $k_{restart} = 2$

The best average solution quality is obtained by setting the cooling ratio to 0.995, the start acceptance ratio to 0.5 and the stopping criteria to 100. This setting also has a reasonably low variance of results compared to other settings.

Team	B6	B7	B8	B9	B10	X11	X12	X13	X14	X15	Score
J06 (corrected)	**2.45**	**0.81**	**1.04**	**1.23**	**1.66**	**1.30**	**1.00**	**1.44**	**1.70**	**0.43**	**13.04**
S21	4.14	3.09	6.30	6.29	3.70	3.13	1.58	5.41	4.29	1.28	39.21
J08	3.85	6.14	16.94	23.83	9.19	2.85	2.66	2.48	5.15	4.03	77.12
S14	11.55	9.48	13.13	25.82	12.66	10.37	11.13	12.85	18.03	14.88	139.89
S23MT	1.54	0.82	1.28	2.29	1.24	TO	0.78	INF	TO	**0**	202.36
S17	19.44	25.17	39.71	56.68	34.47	16.82	23.14	TO	20.87	20.53	297.19
S24	**0**	0.14	**0**	1.06	**0**	**0**	0.06	**0**	**0**	INF	299.96
J06	**2.50**	**0.96**	**1.94**	**2.12**	**3.11**	**1.46**	**1.11**	**2.91**	**INF**	**INF**	**407.5**
S04	7.68	6.41	15.64	26.80	8.58	TO	6.50	INF	14.01	TO	486.04
S22	0.59	0.06	0.18	**0**	0.33	INF	0.01	TO	ERR	ERR	494.29
S08	INF	17.13	24.90	24.68	40.22	10.20	6.50	20.18	13.85	INF	495.24
J05	3.34	3.43	339.67	162.91	22.43	3.79	1.52	3.35	14.35	19.41	574.20
S23	1.54	0.82	1.33	2.28	1.30	TO	TO	INF	TO	TO	624.03
S16	3.44	INF	87.05	47.31	7.15	INF	2.59	TO	TO	INF	773.51
S10	7.39	66.42	3685.91	4779.61	93.76	17.94	15.98	TO	46.34	48.21	8801.92
S10MT	7.49	66.42	3685.91	4779.61	93.76	30.69	61.82	TO	46.34	149.35	8961.75
J16	11.10	12.66	TO	1845.78	12.20	7.70	8.09	TO	12.81	8.11	9330.63
S11	9.47	6.26	1902.45	MEM	INF	6.52	7.41	MEM	11.42	INF	12029.32
S22MT	0.60	**0**	INF	INF	INF	INF	**0**	TO	ERR	ERR	17612.27
S25	CRA	CRA	CRA	CRA	CRA	CRA	CRA	CRA	CRA	CRA	19907.04

Table 3 Percentage wise deviation from best known solution for all teams participating in the competition as well as our improved program. Our team is J06. *INF* means that the found solution is infeasible. *CRA* means that the program crashed. *MEM* means that the program ran out of memory. *ERR* means that the format of the solution is invalid. *TO* means that the program did not finish in time.

6.5 Results

Table 3 shows, for all teams participating in the competition, the percentage wise deviation from best known solution for each instance. The last column shows the teams' official final score which determined the outcome of the competition[5]. This score is the sum of all ten percentages. If a team was unable to find a feasible solution for an instance, their score for this instance was set to twice the objective value of the worst found solution.

Our team identifier is J06 which is ranked seventh in the table. The first row in the table shows the objective values obtained by our program after fixing the bug in the modulation procedure. The bug happened when calculating the allowed modulation in a production campaign, where we did not account for the size of a time step in one special case, this yielded infeasible solutions for X14 and X15. The results from the table show that our new corrected program would have won competition.

The difference in solution quality between our corrected and uncorrected version of the algorithm is due to the use of modulation per scenario described in Section 5.2.

7 Conclusion

We have developed a solver for a large-scale real-life optimization problem using a combination of CP and greedy and local search heuristics.

An initial solution to the complex scheduling problem is found by CP using approximated constraints for production levels and fuel consumption. From this first schedule we apply a stochastic local search algorithm based on a simple neighborhood structure with a fast, but approximative evaluation of the objective function. In the third and final phase we use a greedy algorithm to remove any overproduction.

The solutions obtained are competitive when compared to those found by other teams participating in the final evaluation of the ROADEF/EURO Challenge 2010. After fixing an implementation bug, our approach is robust in the sense that it is always able to find a feasible solution, and it achieves overall the best score, ranking first in the assessment procedure of the competition.

Acknowledgements We would like to thank Marco Chiarandini and Mette Gamst for reading early versions of this paper and providing numerous suggestions for improvement and clarification. We would also like to thank the reviewers for their helpful comments.

We would also like to thank Kristian Schmidt for his help in the qualification phase.

Niels Kjeldsen would like to thank DONG Energy Power A/S for supporting this research.

[5] The results can been seen at `http://challenge.roadef.org/2010/en/results.php`

References

Apt KR (2003) Principles of constraint programming. Cambridge university press

Bisaillon S, Cordeau J, Laporte G, Pasin F (2009) A large neighbourhood search heuristic for the aircraft and passenger recovery problem. 4OR: A Quarterly Journal of Operations Research pp 1–19

Blum C, Blesa MJ, Roli A, Sampels M (eds) (2008) Hybrid Metaheuristics, Studies in Computational Intelligence, vol 114. Springer, DOI 10.1007/978-3-540-78295-7

Burke E, Smith A (2000) Hybrid evolutionary techniques for the maintenance scheduling problem. IEEE Transactions on Power Systems, 15(1):122 –128, DOI 10.1109/59.852110

Chowdhury BH, Rahman S (2002) A review of recent advances in economic dispatch. Power Systems, IEEE Transactions on 5(4):1248–1259

Dunning DJ, Lockfort S, Ross QE, Beccue PC, Stonebraker JS (2001) New York power authority uses decision analysis to schedule refueling of its Indian point 3 nuclear power plant. Interfaces 31(5):121–135

Fourcade F, Johnson E, Bara M, Cortey-Dumont P (1997) Optimizing nuclear power plant refueling with mixed-integer programming. European Journal of Operational Research 97(2):269–280

van Hentenryck P, Milano M (2011) Hybrid Optimization, The Ten Years of CPAIOR, Optimization and its applications, vol 45. Springer, DOI 10.1007/978-1-4419-1644-0

Hooker JN (2011) Hybrid modeling. In: Pardalos PM, van Hentenryck P, Milano M (eds) Hybrid Optimization, Optimization and Its Applications, vol 45, Springer New York, pp 11–62, DOI 10.1007/978-1-4419-1644-0_2

Kim H, Hayashi Y, Nara K (1997) An algorithm for thermal unit maintenance scheduling through combined use of ga, sa and ts. Power Systems, IEEE Transactions on 12(1):329 –335, DOI 10.1109/59.574955

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671

Marwali M, Shahidehpour S (1999) Long-term transmission and generation maintenance scheduling with network, fuel and emission constraints. Power Systems, IEEE Transactions on 14(3):1160 –1165, DOI 10.1109/59.780951

Mukerji R, Merrill H, Erickson B, Parker J, Friedman R (1991) Power plant maintenance scheduling: optimizing economics and reliability. Power Systems, IEEE Transactions on 6(2):476 –483, DOI 10.1109/59.76689

Neumann K, Schwindt C, Zimmermann J (2003) Project scheduling with time windows and scarce resources. Springer

Porcheron M, George A, Juan O, Simovic T, Dereu G (2010) Challenge ROADEF/EURO 2010: A large-scale energy management problem with varied constraints. `http://challenge.roadef.org/2010/index.en.htm`

Satoh T, Nara K (1991) Maintenance scheduling by using simulated annealing method [for power plants]. Power Systems, IEEE Transactions on 6(2):850 –857, DOI 10.1109/59.76735

Schaefer T (1978) The complexity of satisfiability problems. In: Proceedings of the tenth annual ACM symposium on Theory of computing, ACM, pp 216–226

Silva E, Morozowski M, Fonseca L, Oliveira G, Melo A, Mello J (1995) Transmission constrained maintenance scheduling of generating units: a stochastic programming approach. Power Systems, IEEE Transactions on 10(2):695 –701, DOI 10.1109/59.387905

Yellen J, Al-Khamis T, Vemuri S, Lemonidis L (1992) A decomposition approach to unit maintenance scheduling. Power Systems, IEEE Transactions on 7(2):726 –733, DOI 10.1109/59.141779