RAIRO-Oper. Res. 47 (2013) 481–500
DOI: 10.1051/ro/2013053

RAIRO Operations Research
www.rairo-ro.org

A HYBRID APPROACH COMBINING LOCAL SEARCH AND CONSTRAINT PROGRAMMING FOR A LARGE SCALE ENERGY MANAGEMENT PROBLEM *

Haris Gavranović[1] and Mirsad Buljubašić[2]

Abstract. This paper presents a heuristic approach combining constraint satisfaction, local search and a constructive optimization algorithm for a large-scale energy management and maintenance scheduling problem. The methodology shows how to successfully combine and orchestrate different types of algorithms and to produce competitive results. We also propose an efficient way to scale the method for huge instances. A large part of the presented work was done to compete in the ROADEF/EURO Challenge 2010, organized jointly by the ROADEF, EURO and Électricité de France. The numerical results obtained on official competition instances testify about the quality of the approach. The method achieves 3 out of 15 possible best results.

Keywords. constraint satisfaction, local search, optimization, scheduling, ROADEF challenge.

Mathematics Subject Classification. 90-08, 68T20, 90C10, 90C59.

1. Introduction

In this work we take the perspective of a large utility company, tackling their problems in modeling and planning production assets, *i.e.*, a multitude of power

Received October 7, 2013. Accepted October 15, 2013.

* *Special thanks are addressed to the two anonymous Reviewers as well as the Editor for valuable comments and helpful suggestions to improve the quality of the paper.*

[1] Faculty of Engineering and Natural Sciences, International University of Sarajevo, ulica Hrasnička 15, 71210 Sarajevo, Bosnia and Herzegovina. haris.gavranovic@gmail.com

[2] Ecole des Mines d'Ales, LGI2P Research Center, 69 avenue Parc scientifique Georges Besse, 30035 Nimes, France. mirsad.buljubasic@mines-ales.fr

Article published by EDP Sciences

plants. The goal is to fulfill the respective demand of energy over a time horizon of several years, while minimizing a total operation cost.

This problem was proposed at the ROADEF/EURO Challenge 2010, a competition organized by the French Operational Research and Decision Support Society (ROADEF) and the European Operational Research Society (EURO). The problem is defined by Électricité de France (EDF) and it is a real world industry problem solved at EDF.

The proposed problem consists of modeling the production assets and finding an optimal outage schedule that includes two mutually dependent and related subproblems:

(1) Determining the schedule of plant outages. This schedule must satisfy some constraints in order to comply with limitations on resources, which are necessary to perform refueling and maintenance operations.
(2) Determining an optimal production plan to satisfy demand. Production must also satisfy some technical constraints.

The production plan should be determined for different scenarios. The objective is to minimize the expected cost of production.

We present a hybrid heuristic approach to solve the problem. It combines several solution methods: greedy algorithm, constraint programming and local search. The paper focuses on the orchestration of these strategies in order to tackle this problem. A simple reduction of the problem is used in order to solve the biggest instances of the problem more efficiently. This reduction is based on an accurate analysis of the problem structure and consists of decreasing the size of the instances.

The paper is organized as follows. Formal description of the problem is given in Section 2. Section 4 introduces several notations and simplifications, while giving an idea how they would be useful. Section 5 outlines the proposed algorithm explaining its coarse structure first. Thereafter, in the same section, we describe in detail the most important sub-procedures that make the whole algorithm. Section 6 presents the numerical results for the given ROADEF/EURO Challenge instances and compares them with the best known, public results.

2. Problem statement

Here we give a formal definition of the problem. Large part of the section is borrowed from the challenge subject [1] and Godskesen *et al.* [4].
A production portfolio with two types of production units is considered:

- Type-1 power plants which can be supplied in fuel continuously.
- Type-2 power plants which have to be supplied in fuel at regular intervals.

Production assets are used to satisfy a customer demand over a specific time horizon. This time horizon has been discretized with a homogeneous time step. The series of consecutive time steps is called a week. Week length is constant in

each data set. Customer load is uncertain and known only through an available set of uncertainty scenarios. Optimizing for several realistic scenarios instead of just one generally leads to more robust plans.

Each Type-2 plant goes through a number of cycles. A cycle is composed of an outage followed by a production campaign. During the outage, the plant cannot produce electricity. During the production campaign, the plant is able to produce electricity. The maximum number of cycles for each Type-2 power plant is equal to six in all instances proposed by EDF.

Decisions concerning Type-2 plants, *i.e.* scheduling of outages and refueling amounts are shared by all scenarios, but production levels of all plants are determined for each individual scenario.

A production plan specifies the production level of each plant for every time step and every scenario. Furthermore, a maintenance plan specifies when outages of Type-2 plants are scheduled and the amount of fuel to reload at each outage. The objective is to satisfy the demand for electricity at the lowest average cost over all scenarios while satisfying a number of constraints.

Production at a Type-1 plant incurs a cost that is proportional to the power output and depends on the scenario and the time step. Refueling of a Type-2 plants leads to costs proportional to the amount of loaded fuel. The quantity of fuel available for each Type-2 power plant at the beginning of the time horizon is known.

2.1. Variables and bounds

We will use the following indices: $s = 0, \ldots, S-1$ for scenarios, $t = 0, \ldots, T-1$ for time steps, $h = 0, \ldots, H-1$ for weeks, $j = 0, \ldots, J-1$ for Type-1 plants, $i = 0, \ldots, I-1$ for Type-2 plants, and $k = 0, \ldots, K-1$ for cycles.

The following decision variables make up the problem:

- dates for outages to refuel Type-2 plants;
- the amount of fuel that is supplied whenever a Type-2 plant is refueled;
- production levels for Type-1 and Type-2 plants for each time step and scenario.

Let $p(l, t, s)$ denote the production of plant (which may be of type 1 or 2) at time step t in scenario s. Let $PMIN_j^{t,s}$ and $PMAX_j^{t,s}$ denote the minimum and maximum allowed production of Type-1 plant j at time step t in scenario s, then

$$\forall t, s \qquad PMIN_j^{t,s} \leq p(j, t, s) \leq PMAX_j^{t,s}. \tag{2.1}$$

The bounds on production for a Type-2 plants are more complex, since they depend on the current fuel stock and will be defined in the constraints section.

We will denote by (i, k) the k'th outage of Type-2 power plant i. Let $ha(i, k)$ denote the starting week of outage (i, k), and $TO_{i,k}$ and $TA_{i,k}$ denote the earliest and latest possible starting week. Then, we have

$$TO_{i,k} \leq ha(i, k) \leq TA_{i,k}. \tag{2.2}$$

If bounds $TO_{i,k}$ and $TA_{i,k}$ are undefined, the corresponding inequality is trivially satisfied. If outage (i,k) is not scheduled then $ha(i,k) = -1$ and constraint (2.2) is not enforced. Outage $(i,k+1)$ cannot start before outage (i,k) is finished.

The amount of fuel reloaded in outage (i,k) is denoted by $r(i,k)$ and must satisfy the following inequality if (i,k) is scheduled

$$RMIN_{i,k} \leq r(i,k) \leq RMAX_{i,k}. \tag{2.3}$$

If (i,k) is not scheduled, then $r(i,k) = 0$.

In addition to the decision variables, we define a number of auxiliary variables used to represent constraints of the problem.

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 k, the production $p(i,t,s)$ of plant i must be zero for every $t \in ea(i,k)$.

The fuel stock of Type-2 plant i at time step t is denoted by $x(i,t,s) \geq 0$. The initial fuel level of plant i (at time step 0) XI_i is specified in the input data. During the 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:

$$x(i,t+1,s) = x(i,t,s) - p(i,t,s) \times D. \tag{2.4}$$

During an outage, the fuel level at Type-2 plant 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 (i,k), then the new fuel level for i in scenario s is computed using

$$x(i,t+1,s) = Q_{i,k} \times x(i,t,s) + r(i,k) + Q'_{i,k}, \tag{2.5}$$

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

2.2. Constraints

The constraints can be divided into three groups, namely production level constraints, fuel level constraints and outages scheduling constraints.

2.2.1. Production level constraints

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

$$\forall s,t : \sum_{j=0}^{J-1} p(j,t,s) + \sum_{i=0}^{I-1} p(i,t,s) = DEM^{t,s}. \tag{2.6}$$

The bounds on production for Type-1 plants are defined in (2.1). The bounds on production for Type-2 plants depend on the current fuel stock of the plant. If

the fuel level is above a given threshold $BO_{i,k}$ that depends on the production campaign k, then the production is bounded from above by $PMAX_i^t$

$$\forall s,t,i,k : t \in ec(i,k) \wedge x(i,t,s) \geq BO_{i,k} \Rightarrow .0 \leq p(i,t,s) \leq PMAX_i^t. \quad (2.7)$$

As long as the fuel level is above the threshold, there is no lower bound on the production of Type-2 plants in each individual time step, but 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:

$$\forall s,i,k : \sum_{t \in ec(i,k) \wedge x(i,t,s) \geq BO_{i,k}} \left(PMAX_i^t - p(i,t,s)\right) \times D \leq MMAX_{i,k}. \quad (2.8)$$

If the fuel level is below the threshold, the upper bound decreases and a lower bound is also enforced. We will call this the *decreasing power profile (DPP)*. The amount of bound decrease for Type-2 plant i in production campaign k is specified by a decreasing function $PB_{i,k}$ which maps fuel level to a real number in $[0,1]$. 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 $P_x = PB_{i,k}(x(i,t,s)) \times PMAX_i^t$. However, if the plant will run out of fuel if it produces at P_x, it cannot produce at all.

$$(1-\epsilon) \times P_x \leq p(i,t,s) \leq (1+\epsilon) \times P_x \quad (2.9)$$

where ϵ is a small positive real value given in input data.

2.2.2. Fuel level constraints

There are upper bounds on the fuel level before and after the outage. Let $AMAX_{i,k}$ denote the upper bound on the fuel level at the starting time step of outage (i,k) and $SMAX_{i,k}$ the upper bound on the fuel level after outage (i,k). If (i,k) starts at time step t, then the following inequalities must hold for every scenario s:

$$\begin{aligned} x(i,t,s) &\leq AMAX_{i,k} \\ x(i,t+1,s) &\leq SMAX_{i,k}. \end{aligned} \quad (2.10)$$

2.2.3. Outages scheduling constraints

There are several types of temporal constraints between outages that are given in the original problem definition from ROADEF (see CT14-CT18 in [1]). However, all these constraints can be converted to the following constraint:

If a specified pair of outages (i,k) and (i',k') is scheduled such that they both intersect a specified interval (this interval may be the entire planning horizon), then constraint (2.11) must be satisfied.

$$ha(i,k) - ha\,(i',k') \geq S_e \vee ha\,(i',k') - ha(i,k) \geq S'_e, \quad (2.11)$$

where the bounds S_e and S'_e are input data.

For every week h, there is a collection of subsets of outages and for each subset A in this collection, an associated natural number N. For every A and N, at most N of the outages in A are allowed to contain week h

$$\sum_{(i,k)\in A} F(i,k,h) \leq N, \tag{2.12}$$

where $F(i,k,h)$ equals 1 if outage (i,k) is active in week h and 0 otherwise.

There are limited resources available for carrying out maintenance. Thus, a collection of subsets of outages is given. Each subset A in this collection has an associated resource availability Q. For every A and Q, at most Q of the outages in A can use resources in any week

$$\forall h : G(i,k,h) \leq Q, \tag{2.13}$$

where $G(i,k,h)$ equals 1 if outage (i,k) uses resources 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 active.

Finally, there is an upper bound on the capacity that is allowed to be offline at the given time. Thus, a collection of subsets of outages 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 IT, during any week h in IT the total offline capacity of plants corresponding to C cannot exceed $IMAX$

$$\forall h \in IT : \forall t \in h : \sum_{i\in C:\exists k: t\in ea(i,k)} PMAX_i^t \leq IMAX. \tag{2.14}$$

Note that in (2.14) 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.

2.3. Objective function

The objective function is composed of three terms: the total cost of reloading all Type-2 plants, the average cost of Type-1 production over all scenarios, and the average price of remaining fuel at Type-2 plants at the end of the planning horizon over all scenarios.

Let $C_{i,k}$ denote the refueling cost for Type-2 plant i in cycle k, $C_{j,t,s}$ the production cost for Type-1 plant j at time step t in scenario s, and C_i the cost of fuel for Type-2 plant i at the end of the planning horizon. Then the objective function to be minimized is

$$\sum_{i=0}^{I-1}\sum_{k=0}^{K-1} C_{i,k} r(i,k) + \frac{1}{S}\sum_{s=0}^{S-1}\sum_{t=0}^{T-1}\sum_{j=0}^{J-1} C_{j,t,s} p(j,t,s) D - \frac{1}{S}\sum_{s=0}^{S-1}\sum_{i=0}^{I-1} C_i x(i,T,s). \tag{2.15}$$

TABLE 1. The table shows the size of instances A, B and X.

	Instances			
set	J	I	T	S
A	11–31	10–28	1750	10-30
B	19–27	48–56	5565–5817	50–121
X	19–27	48–56	5565–5817	50–121

2.4. INSTANCES

The instances used in this paper are the ones provided by EDF and used at ROADEF/EURO Challenge 2010. All the instances are publicly available and can be found at "http://challenge.roadef.org/2010/en/". They are composed of 3 data sets, A, B and X, each of them containing five instances. Instances in set A are smaller than instances in B and X and are used for the qualification phase of the competition. Instances B and X are similar and about the same size. The size of the instances is illustrated in Table 1.

3. RELATED WORK

A similar problem was studied in Fourcade *et al.* (1997) [8], but with only one scenario and one time step per week. The instances of the challenge are much bigger, and several technical constraints that appear in the challenge version do not appear in Fourcade *et al.*

The method presented in this paper is similar to a hybridized local search and constraint programming approach by Khemmoudj *et al.* [9] but our problem is more complex and more difficult than the one solved in [9].

Several solution methods were designed and tested specifically for the model and the data of the challenge. Pure local search method is proposed by Gardi and Nouioua [2]. The authors achieve the best numerical results reported in the literature and their high-quality solutions are obtained in just a few minutes of computational time. A column Generation approach is proposed by Rozenknop *et al.* [3]. The method achieves the second best results reported in the literature. Godskesen *et al.* [4] present an approach combining local search and constraint programming. Reported solutions are around 1.3% above the best known solutions. The solutions reported in [3] and [4] are numerically comparable with our solutions. A constraint programming approach is proposed by Brandt *et al.* [5]. The same paper also proposes lower bound computations. Jost and Savourey [6] propose a 0–1 integer linear programming approach to solve the problem. The results obtained are comparable to the results in Brandt *et al.* and Lusby *et al.* [7] use the method based on Benders decomposition. The reported implementation is good on small instances (set A) but not competitive on large scale instances (B and X).

4. Notation, assumptions and reductions

In this section, we introduce some new notations and simplifications of the problem that proved to be helpful either for the presentation itself or for improving the quality and the efficiency of the solution.

4.1. Assumption on production costs

The objective function consists of three parts that in a more or less obvious and direct way influence each other. For instances A the part of the cost associated with the production cost for Type-1 plants differs by an order or two of magnitude from the part associated with the fuel cost of Type-2 plants. In bigger instances B and X, these two costs are comparable. As the cost associated with the nuclear plants (Type-2 plants) in most of the given instances is smaller than the cost incurred with Type-1 plants and in a real life the nuclear energy is less expensive than the energy from conventional energy plants, we assume that Type-2 power plants are in general cheaper than Type-1 power plants. This assumption is used throughout the production planing algorithm.

4.2. Preprocessing

We perform simple preprocessing of input data in order to simplify the presentation and the implementation of the solution. Namely, Type-1 power plants have a minimum production level for each time step in each scenario (see inequality (2.1)). The problem is simplified by decreasing the customer demand in each time step and scenario by the sum of these values over all Type-1 power plants while setting the new lower bound on production to zero. Formally,

$$\begin{aligned} \forall s,t : DEM^{t,s} &= DEM^{t,s} - \sum_{j=0}^{J-1} PMIN_j^{t,s} \\ \forall j,s,t : PMIN_j^{t,s} &= 0, PMAX_j^{t,s} = PMAX_j^{t,s} - PMIN_j^{t,s}. \end{aligned} \tag{4.1}$$

This alteration of input data obviously does not change the structure of the initial problem.

4.3. Marginal cost associated to the time step

In order to have a smaller cost, demand should be distributed among the cheapest power plants. Since Type-2 power plants are, by assumption, cheaper, their production levels should be maximized. Nevertheless, for some time steps and scenarios, one part of the demand has to be distributed among Type-1 power plants. The problem of determining the production on Type-1 power plants, given the total production plan for the set of Type-2 power plants, gives rise to a very simple optimization problem similar to the relaxed knapsack problem. The solution consists of sorting the plants according to their respective production cost and

fully employing them one by one, until the demand is satisfied. We will denote by *marginal cost* the cost of the last employed power plant. Formally, the marginal cost at time step t for scenario s is defined as:

$$MC_{t,s} = \max_{p(j,t,s)>0} C_{j,t,s}.$$

We will define the marginal cost at time step t as the average marginal cost over all scenarios:

$$MC_t = \frac{1}{S} \sum_{s=0}^{S-1} MC_{t,s}.$$

It is obvious that if the demand would increase (decrease) by one in a given time step, then the total cost of production would increase (decrease) by the marginal cost associated to that time step. Marginal costs showed to be useful in guiding local search and constraint programming, which will be explained later.

4.4. Decreasing the problem size

One of the things that make this problem very hard is the size of its instances. As the problems may contain a huge number of variables (up to tens of millions), it is an advantage both with respect to computational time and memory consumption to reduce the problem size. We propose a simple way to reduce the size of the problem while keeping some of its original features. The reduction consists of shrinking several consecutive time steps into one. Most of the constraints on the set of outages are given in terms of weeks, and thus it is important to respect their structure in newly created, reduced instances. Therefore, the simplest reduction would be to shrink all time steps from one week, which is described bellow.

Let W denote the week length (number of days in week) in the original instance. The number of time steps in the reduced instance is equal to H, *i.e.* equal to the number of weeks in original instance. Demands, production level bounds, production costs and time step duration of reduced instance are set in the following way:

$$\begin{aligned}
&t \in 0, \ldots, W-1 \\
&\forall s,t : (DEM^{t,s})' = \frac{1}{W} \sum_{m=tW}^{(t+1)W} DEM^{m,s} \\
&\forall s,t,j : (PMAX_j^{t,s})' = \frac{1}{W} \sum_{m=tW}^{(t+1)W} PMAX_j^{m,s} \\
&\forall s,t,i : (PMAX_i^{t})' = \frac{1}{W} \sum_{m=tW}^{(t+1)W} PMAX_i^{m} \\
&\forall s,t,j : (C_{j,t,s})' = \frac{1}{W} \sum_{m=tW}^{(t+1)W} C_{j,m,s} \\
&D' = W \times D.
\end{aligned} \tag{4.2}$$

TABLE 2. The table compares the size and the cost for shrinked and original instances. The second and the third column represent the size in megabytes of data files used to store original and reduced instance. The fourth column represents the solution cost of original instance obtained by using the outages schedule of the best solution of the shrinked instance whose cost is given in the last column.

Shrinked instances				
Instance	Original size (MB)	Shrink size (MB)	Original cost (10^{10})	Shrink cost (10^{10})
dataB6	139.9	5.6	8.41	8.13
dataB7	144.3	5.6	8.27	7.93
dataB8	262.0	10.3	8.43	8.11
dataB9	262.0	10.3	8.82	8.37
dataB10	251.7	9.8	7.99	7.51
dataX11	140.0	5.5	8.01	7.71
dataX12	143.2	5.5	7.87	7.50
dataX13	262.1	10.4	7.83	7.42
dataX14	262.1	10.4	7.85	7.42
dataX15	249.8	9.7	7.68	7.09

Keeping all other parameters like fuel costs, fuel stock bounds, refueling bounds and outages constraints the same, even the values of objective functions of original and shrinked instances turned out to be comparable and fairly close. As an example, for an instance having 250 weeks, 1750 time steps (week length = 7) and time step duration 24, the shrink of a week into one time step results into an instance with 250 weeks, 250 time steps and time step duration 168. Shrinked instances are much smaller and easier to solve. Obtaining the first feasible solution for some of the instances in sets B and X without this reduction showed to be almost impossible in a reasonable computing time. In Table 2, we compare the size and the cost of shrinked instances to the original ones.

5. SOLUTION

In this section, we describe the heuristic method for solving the proposed problem. We first give a general description of the method, followed by detailed description of solution components.

5.1. GENERAL METHOD

The approach combines constraint satisfaction, local search and a greedy constructive optimization algorithm. The simple procedure described in Section 4.4 is used to decrease the problem size for huge instances. The proposed problem includes two dependent and related subproblems: determining the schedule of outages and determining an optimal production plan to satisfy the demand. Solving

both subproblems together seems to be almost impossible since the number of decision variables can be huge and the existence of nonlinear constraints makes it even more difficult. Therefore, a feasible solution to the problem is obtained by solving these two subproblems in a consecutive manner. First, a feasible outage schedule is determined, followed by a production plan compatible with this schedule. The feasible solution found this way is then improved using a series of local improvements. The newly obtained locally optimal solution gives rise to a new outages scheduling phase and the cycle repeats. This would be, in short, a description of the solution we propose here. A constraint Programming (CP) approach is used to solve the problem of scheduling outages. A greedy constructive procedure determines a feasible production plan for a given schedule, then a local search approach is used to improve the solutions. A detailed description of these three main solution blocks is given in the following three subsections. The procedure combining these three blocks is given in Algorithm 5. After this, we describe the use of a shrink procedure for scaling the problem for huge instances and give the final algorithm.

5.2. Outages scheduling

Different approaches can be used to solve the outages scheduling problem, such as local search, constraint programming or constructive procedure, but using a commercial constraint programming solver emerged to be the most reliable, and probably the most elegant approach, with very good results.

Since the exact problem representation would result in a huge number of variables, we focus only on finding a feasible maintenance schedule, leaving the rest of the optimization to the subsequent local search.

All outage scheduling constraints, *i.e.* constraints of type (2.2) and those of type (2.11)–(2.13) are modeled precisely. The resulting CP model, without objective function, for this scheduling problem is given below. We use the notation defined in Section 2.

$$\begin{aligned} TO_{i,k} \leq ha(i,k) \leq TA_{i,k} \\ ha(i,k) - ha(i',k') \geq S_e \vee ha(i',k') - ha(i,k) \geq S'_e \\ \sum_{(i,k)\in A} F(i,k,h) \leq N \\ \forall h : G(i,k,h) \leq Q. \end{aligned} \tag{5.1}$$

The constraints are modeled using the OPL Optimization Programming Language offered within IBM Ilog suite, which provides basically all necessary constructions to efficiently and easily represent every single constraint.

We have solved the same scheduling problem also with other CP solvers such as Comet and Mistral (see [14] and [15]). They were all capable to efficiently solve all official instances. The solution time varies from a fraction of a second up to several seconds for smaller data sets and a few minutes on larger ones. The resulting schedule is valid with respect to the scheduling constraints. On the one hand,

the obtained solutions for outages scheduling suffer from a lack of diversification, and on the other hand, from the fact that these schedules do not always allow a feasible production plan. This is why further constraints are added to the model to ensure the feasibility of the remaining production assignment problem. A certain amount of fuel has to be consumed in every production campaign to reach the fuel level limits that apply before or after the upcoming refueling (see constraint (2.10)). Thus, based on the existing outages schedule and an approximate unfeasible production plan, necessary spacing constraints are calculated and added between every two successive outages. The approximate production plan is based on minimum refueling and maximum power strategy. Namely, for the given schedule, we set the minimum refueling amount for each outage (i, k) and the maximum production level for each Type-2 power plant in each time step not in outage. If fuel levels calculated this way violate constraint (2.10), we say that the schedule is unfeasible. It is obvious that these unfeasible schedules are also unfeasible for every possible production plan, *i.e.* they don't allow any feasible production plan. On the other hand, a feasible schedule with respect to the approximate production plan is not necessarily feasible for a production plan respecting all the constraints. Every solution with constraint (2.10) violated mandates the addition of a new set of increased spacings between outages and the model is solved again. This quest for feasibility comes at a cost, but the number of repeats is usually small and ranges from 1 to 5 on the provided set of instances.

5.2.1. Improved scheduling

The outages scheduling could be optimized from the very beginning of the solution procedure. The marginal cost introduced in 4.3 and the numerical experiments endorse the scheduling of outages in the weeks of low demand over the scheduling in weeks of high demand. The solver is guided to these solutions by using a set of weeks of low demand as its starting search point. The week demand is defined as the sum of demands over all time steps in a week and over all scenarios. Formally, the starting point $SP(i, k)$ for each outage (i, k) is set as follows:

$$\begin{aligned} &\forall h : DEM(h) = \sum_{s=0}^{S-1} \sum_{t \in h} DEM^{t,s} \\ &\forall i, k : SP(i, k) = h' \Leftrightarrow DEM(h') = \min_{h \in [TA_{i,k}, TO_{i,k}]} DEM(h). \end{aligned} \tag{5.2}$$

This simple heuristic is used exclusively in the beginning, when no feasible solution for the problem is available. It does not come as a surprise that this guided search shows better numerical results than randomized solutions for the outage schedule problem. In the advanced stages of search, when feasible or even good solutions are available, the weeks with the smallest marginal cost are used as the starting

point for the solver, *i.e.*

$$\forall h : MC(h) = \sum_{s=0}^{S-1} \sum_{t \in h} MC_{t,s}$$
$$\forall i,k : SP(i,k) = h' \Leftrightarrow MC(h') = \min_{h \in [TA_{i,k}, TO_{i,k}]} MC(h). \tag{5.3}$$

The outages scheduling procedure is given in Algorithm 1.

Algorithm 1 *outagesSchedule*()

```
INPUT: startPoint - solver starting point
schedule = solveCSPModel(startPoint) - find outages schedule using solver
Greedy Production Planning - min refuel max power strategy
while constraints violated do
   add spacing constraints
   schedule = solveCSPModel(startPoint)
   Greedy Production Planning
end while
```

5.3. Production assignment

The next step to build a feasible solution is production planning. This part of the solution assigns production levels for all plants and refueling amounts for all outages. For the outages schedule given by the solver, a solution is found by a constructive production setting procedure. This solution is not always feasible since constraint (2.10) cannot always be satisfied. In this case, the additional spacing constraints are added to the CP model and the model is solved again.

The algorithm, in a greedy manner, assigns as much production as possible to the cheapest plants until the demand is covered. By the assumption of Type-2 power plants production cost being smaller than Type-1 plants production cost, it is desirable for Type-2 power plants to produce as much as possible. To achieve this, refueling amounts should be as big as possible while respecting all the constraints. Also, the decrease power profile length should be short, since the maximum allowed power is generally smaller during this profile.

On the other hand, a certain amount of fuel is lost during refueling (see constraint (2.5)) and, therefore, it is not desirable to have "too big" fuel stock before the refueling.

For these reasons, we implement the following refueling strategy for each Type-2 power plant:

- initially choose the minimum refuel amount at each outage;
- increase the refuel amount at the previous outage if the plant enters the decrease profile in the next time step (if increase is possible).

Setting Type-2 plants production levels as described can lead to a problem of overproduction. Overproduction occurs if a total production over all Type-2 power plants extends over the customer demand at a certain time step for a certain scenario. Actually, using the described strategy causes overproduction for all B and X instances. This means that, at certain time steps (usually time steps with relatively small demand), some of Type-2 power plants should not be powered to their maximum. The problem is then to choose some Type-2 power plants and drive them to use the associated modulation on the production.

On the other hand, solving the overproduction problem by using modulation can lead to the violation of maximum modulation constraint (see constraint (2.8)). Therefore, the total modulation necessary to cope with the overproduction problem should be properly distributed among Type-2 plants. Simultaneously, solving the problems of overproduction and overmodulation can lead to violation of constraints imposed on maximum fuel level before (after) the outage (see (2.10)).

The algorithm handles these difficulties by sorting Type-2 power plants for each time step before setting the production. Namely, the algorithm sets the production in ascending order of time steps, and for each time step t, a sorted list of Type-2 power plants is calculated. The production level is then set for each plant in the list as described before, *i.e.* as high as possible with eventually increasing the refuel (satisfying all constraints). Thus, the modulation will usually be used for the last plants in the list.

To avoid the feasibility problems, power plants for which these problems are highly probable are placed on the top of the list. These are the plants with a minimum amount of modulation that can still be used in a current production campaign while satisfying maximum modulation and fuel bounds constraints (constraints (2.8) and (2.10)). Note that the power plants that are in decreasing power profile in a current time step should also be at the top of the list because of the imposed lower bound on production. The sorting procedure is given in Algorithm 3.

The production levels of Type-2 power plants are equal for all production scenarios. We will sometimes use $p(i,t)$ instead of $p(i,t,s)$ to denote the production of Type-2 plant i in any scenario. The remaining demand is distributed among Type-1 plants for all scenarios using the simple procedure mentioned in Section 4.3.

The presented production assignment procedure efficiently finds the initial feasible production plan for all 15 provided benchmarks and the pseudo code is given in Algorithm 2.

5.4. Local improvements

Here we present three local search strategies to optimize the solution. Two strategies optimize the production plan. The first improves the refueling process, while the other one directly seeks for a better power settings of plants. The third one moves outages locally, and thus improves the outages schedule quality. Local improvements are repeatedly used in a given order until no improvements are

Algorithm 2 *setProduction*()

INPUT: outages schedule
set refuel amounts to minimum
for t = 0 to T-1 **do**
  sortType2Plants(t)
  for m = 0 to I - 1 **do**
    $i \leftarrow$ index of m-th plant in the list
    Set $p(i,t)$ as high as possible - respecting demand
    Increase refuel on previous outage (if possible) if plant will enter the imposed power profile in $t+1$
  end for
end for
set Type-1 plants production at time step t for all scenarios

Algorithm 3 *sortType2Plants*(t)

$list_1 \leftarrow$ power plants in DPP at time step t
$list_2 \leftarrow$ power plants not in DPP at time step t
for m = 0 to $list_2.size()$ **do**
  $i \leftarrow$ index of m-th plant in $list_2$
  $k \leftarrow$ index of production campaign of plant i in t
  $CURR\_MOD \leftarrow$ current modulation of plant i in campaign k (before t)
  $MAX\_REM\_MOD_i = MAXM_{i,k} - CURR\_MOD$
  $CURR\_F \leftarrow$ current fuel level of plant i
  $t' \leftarrow$ last time step of production campaign k
  $MAX\_REM\_MOD'_i \leftarrow AMAX_{i,k} - (CURR\_F - \sum_{t''=t}^{t'} PMAX_i^{t''})$
  $MAX\_REM\_MOD_i \leftarrow min(MAX\_REM\_MOD_i, MAX\_REM\_MOD'_i)$
end for
Sort $list2$ in increasing order by $MAX\_REM\_MOD_i$
return $merge(list_1, list_2)$

found or a given time limit is reached. The impact of local improvements on the solution is illustrated in Figure 1.

5.4.1. Local Search based on marginal cost

This procedure tends to improve the feasible solution modifying the production levels of a Type-2 power plant at only two time steps at once. Both time steps should be inside a single production campaign and outside an imposed power profile. The change should not affect the total production in a given campaign, *i.e.* power decrease at one time step should be equal to the power increase at the second time step. These restrictions make the change very simple since refuel amounts and fuel levels at the beginning (end) of the campaign and modulation do not change. To satisfy the demand constraint, adjustment of the production levels of Type-1 power plants at two appropriate time steps should be done. The overall cost of the solution changes only according to the change of the production levels

of Type-1 power plants. Here, the notion of marginal cost proved to be useful, both to find interesting pairs of time steps and to determine the scope of change.

Consider a Type-2 power plant j such that there is a time step t_1 where it is not employed at its maximal power, and it is not in a decreasing profile. Suppose we have another time step t_2 in the same production campaign such that $MC_{t_2} \leq MC_{t_1}$. Obviously, it is profitable to increase the power of j at t_1 by some small δ and decrease the power by δ at t_2. The total cost will decrease by $(MC_{t_1} - MC_{t_2}) \times \delta$. Values for δ are chosen to be small enough to have only one Type-1 plant production change at t_1 and only one Type-1 plant production change at t_2.

The search is done for each Type-2 power plant and each production campaign until no profitable change can be found. The pseudo code is given in Algorithm 4.

Algorithm 4 *LS*1

INPUT : power plant i, production campaign k
sort time steps of campaign k in descending order of marginal cost
while true **do**
  Find time step t_1 outside DPP s.t. $p(i,t_1) < PMAX_i^t \wedge MC_{t1} = max_{t\in k} MC_t$
  Find time step t_2 outside DPP s.t. $p(i,t_2) > 0 \wedge MC_{t2} = min_{t\in k} MC_t$
  if $MC_{t1} \leq MC_{t2}$ **then**
    terminate
  end if
  $p(i,t_1) \leftarrow p(i,t_1) + \delta$
  $p(i,t_2) \leftarrow p(i,t_2) - \delta$
  Update the list
end while

5.4.2. Tuning the refuel levels

It is not obvious how to determine the best refueling amounts. The change in one refuel propagates through all production campaigns and the presence of a decreasing profile makes these changes non-linear. The production assignment procedure sets up initial, satisfactory good amounts of refuel at each outage. These amounts are further optimized with respect to the total cost of the solution using small incremental changes. A random outage (i,k) is chosen and the difference in overall cost for a small amount of refuel change δ is calculated. If the solution improves, the change is made and the process continues. Different, positive and negative, values for δ are chosen. The whole process continues until a local optimum is met, *i.e.* when all pairs (i,k) have been checked.

5.4.3. Local improvements for outages schedule

These local improvements proved to be the most valuable in terms of improving the solution quality. At the same time, the idea to move around one or several outages, exploring the search space, is also the most natural one. This exploration comes at a cost that consists mainly of the evaluation of the improvement

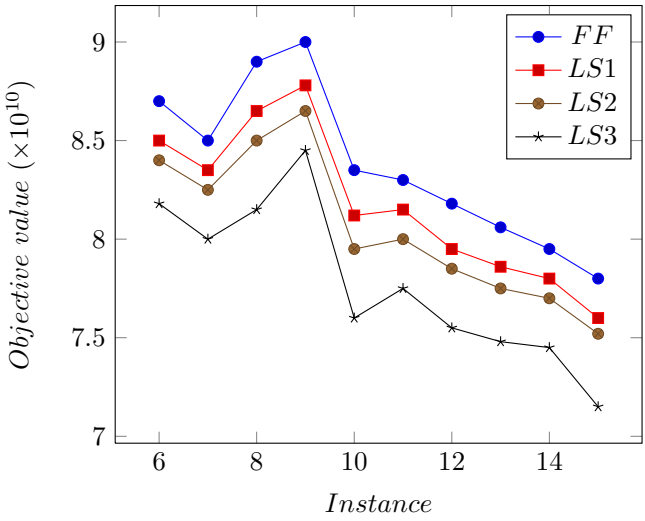

FIGURE 1. Solution improvement for shrinked B and X instances (B6-B10, X11-X15). The blue line represents the first feasible solution (FF) while the red, green and black lines represent the solutions after marginal cost (LS1), tuning fuel levels (LS2) and moving outages (LS3) local improvements respectively.

(or deterioration) of the solution. Changing the start time of the outage will change the fuel levels and production of the current Type-2 plant, as well as production levels of other plants (usually Type-1 power plants). All these changes will affect the overall solution cost. The move and the associated production assignment have to be feasible to be accepted. When there is an improvement, the move is accepted and performed. The moves on at most two outages are examined simultaneously. Creating the whole solution and computing its cost is time consuming. In the deployed solver, the quality of the moves for the outages is estimated using the changes in marginal cost and the estimation of the implied refuel change. In this way, the search becomes efficient and complies with the time constraints imposed by the rules of the challenge.

The order of selection of the outages to be moved influences the final quality of the solution. We tried several strategies, such as choosing the best move, first improving move, and random move. Moving the best of all outages shows the best performance despite the fact that it is not the most efficient in terms of time.

5.5. Scaling – shrink

This section describes the use of shrink procedure for scaling the problem size. The quality of the final solutions and the efficiency of the method for B and X instances are strongly affected by their size. The idea to shrink the instances proves

Algorithm 5 *solve*

```
INPUT 1: dataSet - data set to solve
INPUT 2: solverStartPoint - starting point for CSP solver
if solverStartPoint = NULL then
   solverStartPoint = weeks with low demand
end if
while time limit not exceeded do
   schedule = outagesSchedule(solverStartPoint)
   setProduction(schedule)
   while solution not feasible do
      add spacing constraints
      schedule = outagesSchedule(schedule)
      setProduction(schedule)
   end while
   LocalSearch()
   update best cost
   solverStartPoint = weeks with low marginal cost
end while
```

to be useful to tackle this problem to a certain degree. The basic idea that we use is to solve the reduced instance, and then use the set of obtained outages as a starting point in solving the original instance. The quality of the first feasible solution for the original instance obtained this way showed to be very high. This is why, along with the size of the instances, these solutions could not be significantly improved in reasonable computing time. Therefore, most of the computing time is devoted to solving the reduced instances. This is also the reason why most of the presented experiments illustrated in figures is done for reduced instances. The full method for B and X instances is presented in Algorithm 6. Nevertheless, exactly the same method is used to solve the smaller instances A. The only difference is that these small instances are not shrinked in the pre-processing phase.

Algorithm 6 Final Algorithm

```
INPUT: dataSet - data set to solve
shrinkDataSet = shrinked data set
solve(shrinkDataSet) - solve shrinked data set using algorithm 1
bestOutages = outages from the best solution of shrinked data set
solve(dataSet, bestOutages) - solve the original data set using with bestOutages as a
starting point
```

6. Computational results

The whole solution is implemented in C++ programming language on Linux x86-64 architecture and compiled with GCC version 4.4.3. The solver chosen for the

TABLE 3. The table shows the solutions achieved by our method. We compare the results to the best known results. Results denoted by * are achieved by Gardi *et al.* [2] and the results denoted by † were achieved by the competitors of the ROADEF/EURO challenge.

	Solutions		
Inst	Our Solution	Best Known Solution	Score(%)
A1	169 474 519 241	169 474 800 000*	**-0.0002**
A2	145 956 733 339	145 956 800 000*	**-0.00005**
A3	154 277 239 128	154 316 000 000*	**-0.025**
A4	111 505 728 462	111 494 000 000*	0.010
A5	124 716 680 000	124 543 900 000*	0.128
B6	837 632 963 07	834 247 162 17†	0.405
B7	820 702 010 04	810 997 200 00*	1.196
B8	837 866 683 28	818 997 400 00*	2.301
B9	875 425 269 18	816 895 600 97†	7.164
B10	794 669 685 74	777 670 249 99	2.185
X11	796 508 419 33	790 096 500 00*	0.811
X12	782 742 078 67	775 639 900 00*	0.915
X13	777 210 105 49	762 885 200 00*	1.877
X14	780 275 365 71	761 494 800 00*	2.466
X15	763 107 410 43	743 883 700 00*	2.584

constraint satisfaction part of the problem is IBM ILOG CPoptimizer version 12. All results reported in the paper are obtained on a computer equipped with an Intel i7 920 processor (2.66 GHz, 8M Cache, RAM 6 GB).

The performance was not our main concern, and we believe that the speed of the actual implementation of the proposed algorithm could be improved significantly. The results obtained on 15 official instances used at EURO/ROADEF Challenge 2010 are presented in Table 3. Time limits imposed by ROADEF/EURO Challenge 2010 are respected (30 min for A instances and 60 min for B and X instances). Almost all obtained solutions lie within the range 0.5%–2% of the best known solutions while on the instances A, this gap is substantially smaller. For three instances from set A, we obtain the best known results to our knowledge. It is important to mention that only B and X instances are considered for the final ranking of ROADEF/EURO Challenge 2010. The presented results were obtained after the competition and taking into consideration the possibility that all the teams found feasible solutions, which was not the case at the competition, it's fair to say that the presented method would be ranked in the top five.

7. CONCLUSION

In this paper, we propose a hybrid method to solve a large scale energy management problem. The methodology used to tackle the problem consists of, among

other things, constructive and greedy algorithms, local search procedures and constraint programming techniques. The method first solves two interdependent subproblems in a consecutive manner and constructs a feasible solution. It then proceeds with improvements of the solution applying several local search techniques. The method is implemented in C++ and is fully operational and can be used to solve real world instances. The numerical results are comparable with the best known, for some of them our method finds the solutions of practically the same quality and even improves several best results.

References

[1] M. Porcheron, A. Gorge, O. Juan, T. Simovic and G. Dereu, Challenge roadef/euro 2010: A large-scale energy management problem with varied constraints (2009), http://challenge.roadef.org/2010/files/sujetEDFv22.pdf.

[2] F. Gardi and K. Nouioua, Local Search for Mixed-Integer Nonlinear Optimization: a Methodology and an Application, *EvoCop: 11th European conference on evolutionary computation in combinatorial optimisation*, Torino, Italy (2011).

[3] A. Rozenknop, R. Wolfler Calvo, L. Alfandari, D. Chemla and L. Letocart, Solving the electricity production planning problem by a column generation based heuristic. *J. Scheduling* (2012) doi: 10.1007/s10951-012-0286-9.

[4] S. Godskesen, T.S. Jensen, N. Kjeldsen and R. Larsen, Solving a real-life, large-scale energy management problem. *J. Schedulin* (2012) doi: 10.1007/s10951-012-0279-8.

[5] F. Brandt, R. Bauer, M. Völker and A. Cardeneo, A constraint programming-based approach to a large-scale energy management problem with varied constraints. *J. Scheduling* (2012) doi: 10.1007/s10951-012-0281-1.

[6] V. Jost and D. Savourey, A 01 integer linear programming approach to schedule outages of nuclear power plants. *J. Scheduling* (2013) doi: 10.1007/s10951-013-0322-4.

[7] R. Lusby, L. Muller and B. Petersen, A solution approach based on Benders decomposition for the preventive maintenance scheduling problem of a stochastic large-scale energy system. *J. Scheduling* (2011) 10.1007/s10951-012-0310-0.

[8] F. Fourcade, E. Johnson, M. Bara and P. Cortey-Dumont, Optimizing nuclear power plant refueling with mixed-integer programming. *Eur. J. Oper. Res.* **97** (1997) 269–280.

[9] M. Khemmoudj, M. Porcheron and H. Bennaceur, When constraint programming and local search solve the scheduling problem of Électricité de France nuclear power plant outages, In *CP 2006, the 12th International Conference on Principles and Practice of Constraint Programming*, edited by F. Benhamou. Lecture Notes in Computer Science, vol. 4204. Springer, Berlin, Germany (2006) 271–283.

[10] R.A. Krzysztof, *Principles of constraint programming*, vol. 420. Cambridge University press (2003).

[11] IBM ILOG CPoptimizer and Cplex User's Guide (2009).

[12] M. Milano and P. Van Hentenryck, *Hybrid Optimization*, Springer Optimization and Its Applications (2010).

[13] G. Pesant and M. Gendreau, A Constraint Programming Framework for Local Search Methods. *J. Heuristics* **5** (1999) 255–279.

[14] Comet, http://dynadec.com/technology/constraint-programming/.

[15] Mistral, http://www.cs.wm.edu/~tdillig/mistral/index.html.