Combining Dantzig-Wolfe and Benders decompositions to solve a large-scale nuclear outage planning problem

Rodolphe Griset, Pascale Bendotti, Boris Detienne, Marc Porcheron, Halil Şen, François Vanderbeck

Combining Dantzig-Wolfe and Benders decompositions to solve a large-scale Nuclear Outage Planning Problem

Rodolphe Griset[a,b,*], Pascale Bendotti[a], Boris Detienne[b], Marc Porcheron[a], Halil Şen[b,1], François Vanderbeck[b,2]

[a] *EDF R&D Osiris, EDF LAB Boulevard Gaspard Monge, 91120 Palaiseau, France*
[b] *INRIA team RealOpt, A33, 351 Cours de la Libération, 33400 Talence, France*

Abstract

Optimizing nuclear unit outages is of significant economic importance for the French electricity company EDF, as these outages induce a substitute production by other more expensive means to fulfill electricity demand. This problem is quite challenging given the specific operating constraints of nuclear units, the stochasticity of both the demand and non-nuclear units availability, and the scale of the instances. To tackle these difficulties we use a combined decomposition approach. The operating constraints of the nuclear units are built into a Dantzig-Wolfe pricing subproblem whose solutions define the columns of a demand covering formulation. The scenarios of demand and non-nuclear units availability are handled in a Benders decomposition. Our approach is shown to scale up to the real-life instances of the French nuclear fleet.

Keywords: OR in energy, Nuclear outage scheduling, Integer programming, Dantzig-Wolfe decomposition, Benders decomposition

1. Introduction

Nuclear production is a major source of electricity production for EDF, which operates 58 reactors at 19 locations in France. In the following, reactors are referred to as *nuclear units* and locations as *power plants*. Each nuclear unit must undergo a periodic outage to perform maintenance tasks and to reload nuclear fuel, thus leading to a complex industrial process. In particular, each outage of a nuclear unit must be scheduled in advance to allow for coordination of EDF's personnel and subcontractors. Furthermore, a nuclear outage induces an expensive substitute production by other means, e.g., gas-fired units or

*Corresponding author
Email addresses: rodolphe.griset@edf.fr (Rodolphe Griset), pascale.Bendotti@edf.fr (Pascale Bendotti), boris.detienne@u-bordeaux.fr (Boris Detienne), marc.porcheron@edf.fr (Marc Porcheron), halil@mapotempo.com (Halil Şen), fv@atoptima.com (François Vanderbeck)
[1] Current affiliation: Mapotempo, 11 Allée de la Pacific, 33800 Bordeaux
[2] Current affiliation: Atoptima, 16 Place Sainte-Eulalie, 33000 Bordeaux

purchases on electricity market, to fulfill the electricity demand. For these reasons, scheduling nuclear outages is of major economic importance for EDF.

In this article, we consider a fixed number of outages to be scheduled within a given time horizon. The electricity demand is discretized over the time horizon and includes potential sales in the electricity market. At each time period, demand must be met by available generation units or market purchases. Nuclear outages are subject to *scheduling constraints* caused by limited resource availability as unit refueling and maintenance operations share the same resources in terms of personnel and equipment. Nuclear units must account for various *operational constraints*, which also impact nuclear outages, namely upper-bound on remaining fuel levels at outage starts, limiting operation at intermediate power, non-linear decreasing production profiles once the fuel level falls below a given threshold. As the time horizon is large, the data is subject to uncertainty in particular the demand, prices and capacities of exchanges on the market, but also the availability of non-nuclear units. The Nuclear Outage Planning Problem (NOPP) is to find a minimum cost plan for the nuclear refueling outages satisfying both scheduling and operational constraints to meet the demand in an uncertain future. The uncertainty is modeled by a set of scenarios in a stochastic setting. The NOPP can be formulated as a two-stage decision problem. The first-stage decisions are nuclear outage dates, which have to be fixed before knowing the future; the second-stage decisions correspond to the production plan once uncertainty is revealed.

In the 90's, different approaches were investigated on a deterministic variant of the problem. Edwin and Curtius [6] and Mukerji et al. [14] consider an Integer Linear Programming (ILP) approach in which nuclear decreasing profile constraints are relaxed. Furthermore both studies limit their tests to one nuclear power plant with at most six nuclear units and a one-year horizon.

Fourcade et al. [8] report a high increase in solution-time when attempting to solve a compact ILP formulation on instances involving several power plants over a three-year horizon, while enforcing binding scheduling constraints between outages from units of different power plants. To reduce the solution time, they apply Lagrangian relaxation to the demand constraint. The corresponding decomposition scheme is solved through Uzawa's sub-gradient algorithm [1]. Each sub-problem, corresponding to a nuclear power plant, is solved through a Branch & Bound algorithm which has been enhanced with a clique cut generation of local power plant scheduling constraints. However, this approach presents a computational limitation related to the use of Uzawa's algorithm. It also needs a re-dispatching phase to satisfy the demand, which could not ensure that solutions remain feasible.

In 2010, EDF submitted the stochastic variant of the problem to the academic community through the ROADEF Challenge [16]. In the proposed problem specification, the amount of nuclear fuel to be reloaded was a continuous decision, the time period was from four to six hours and there were up to 500 stochastic scenarios. Moreover, the solution time was limited to one hour. Given these characteristics, the best results were obtained by (meta-)heuristic approaches, thus reinforcing EDF's choice to use such methods in the current operational

solution to the NOPP . However, most of the teams involved in the challenge took advantage of the natural two-stage structure of the problem by embedding an MIP or LP formulation in some specific phase of their solution scheme. To deal with the various scheduling constraints, Jost and Savourey [12] propose an ILP formulation to fix outages dates, whereas Brandt [4], Godskesen et al. [10], Gavranović and Buljubašić [9] apply constraint programming methods. Once first-stage decisions are fixed it is possible to solve the production dispatching problem through a Benders' like reformulation suitable for row generation. In [13], such an approach is used, where specific constraints of the nuclear units are relaxed in a first phase before a so called reparation phase is performed. Rozenknop et al. [17] propose a dedicated state-space graph embedding nuclear operational constraints. A path in the graph corresponds to a feasible production plan with respect to the fuel level and the outage time-window constraints. The set of feasible plans is generated dynamically using a column generation scheme. The principle is that each plan prescribes fixed production levels for each week. The authors report computational issues as a lot of plans have to be generated, and generating each plan is computationally demanding.

Some characteristics did change over time in the definition of the problem since the ROADEF challenge. The amount of fuel to be reloaded was assumed to be variable. This assumption is no longer currently valid as the amount of fuel is fixed in the operational data. Hence the approaches presented in this article account for a fixed amount of fuel, even though they can account for a variable amount. Given the economic value at stake, a solution with guaranteed quality is of particular interest for EDF. To this end, some characteristics and specifications can be slighted amended is set back to a week and the solution time is extended to eight hours. These amendments make it possible to use approaches based on an exact solution procedure, in particular those combining reformulations and decompositions.

The principal focus of the current study is the stochastic aspect of the problem. A preliminary step is to deal with a single scenario in a deterministic setting. The idea is to find a suitable mathematical programming formulation to solve the corresponding NOPP instances at optimum. Then the number of scenarios considered is increased in a stochastic setting. The goal is then to solve NOPP instances with a quality guarantee whenever an exact solution is non-achievable. From an operational point of view the interest is to capture the impact of a given stochastic representation in the guaranteed solution quality.

Aside from ROADEF Challenge, a two-stage extended formulation, proposed in [11], involves a state-space graph in the first stage where an arc corresponds to either a production or an outage period. Our present work is derived from the latter formulation. Such an approach is computationally attractive because it involves network flow subsystems, thus leading to tight formulations. However, the size of the resulting model seriously impairs the scalability of the approach when facing real size instances. Our contribution is to propose an efficient, sparse, extended formulation for real size instances. When the number of scenarios increases, we resort to a double decomposition scheme to solve large-scale instances of the problem. Dantzig-Wolfe decomposition is used to manage

the large-scale instances at first-stage and Benders' decomposition to exploit the independence of the sub-problems associated with different scenarios at second stage. We also study the effect of introducing second-stage variables representing a surrogate measure of the first-stage decisions. The problem is finally solved using a dedicated row-and-column generation algorithm. Our numerical study shows that, in the deterministic setting, real size instances can be solved to optimality directly using a commercial MIP solver with the proposed extended formulation. In the stochastic setting with up to 32 aggregated scenarios built from a 484 scenario database, the proposed row-and-column generation based method provides good quality solutions.

In Section 2, the nuclear outage planning problem is described. In Section 3, variants of the proposed extended formulation are presented. Section 4 gives the details of the Dantzig-Wolfe and Benders reformulations dedicated to handle real size stochastic instances through a row-and-column generation technique. In Section 5, the computational results attest the effectiveness of the proposed approach on real size (deterministic and stochastic) EDF instances of the NOPP. Some concluding remarks end the paper in Section 6.

2. Problem description

The objective of the NOPP variant considered in this paper is to find a plan for the nuclear refueling outages that minimizes, on a given time horizon, the expectation cost of non-nuclear units on several scenarios of demand, market prices and capacities, and fossil unit availability, while satisfying the demand and both the scheduling and operational constraints.

2.1. Stochastic aspects

In its most general form, the NOPP is a multi-stage stochastic problem. The plan of nuclear refueling outages is re-optimized every month, given that the outages scheduled to occur in less than a year are almost fixed. The demand, availability of nuclear and non-nuclear units and costs of non-nuclear units are known weeks and sometimes only days in advance, and unit production is re-optimized up to 30 minutes before producing.

In this article, we assume that the availability of nuclear units is deterministic, while the demand and non-nuclear units are stochastic and subject to uncertainty. The uncertainty is represented through a set of aggregated scenarios built using an EDF library with a database of 484 original scenarios (see Section 5). These assumptions allow for modeling the NOPP as a two-stage problem. In the first stage the outage dates are the here-and-now decisions taken before uncertainty is revealed, while in the second stage, the aim is to have a feasible production plan – i.e., the demand is met and the fuel level constraints of nuclear units are satisfied – which minimizes the leasing cost of non-nuclear units.

In each scenario, fictitious units have been included to guarantee that production can meet the demand. A *failure unit* with a very high cost and infinite power capacity is added to the non-nuclear fleet to prevent any under-capacity of production with respect to the demand. Thus a solution with insufficient nuclear production will be feasible but expensive. Similarly a *load shedding unit*

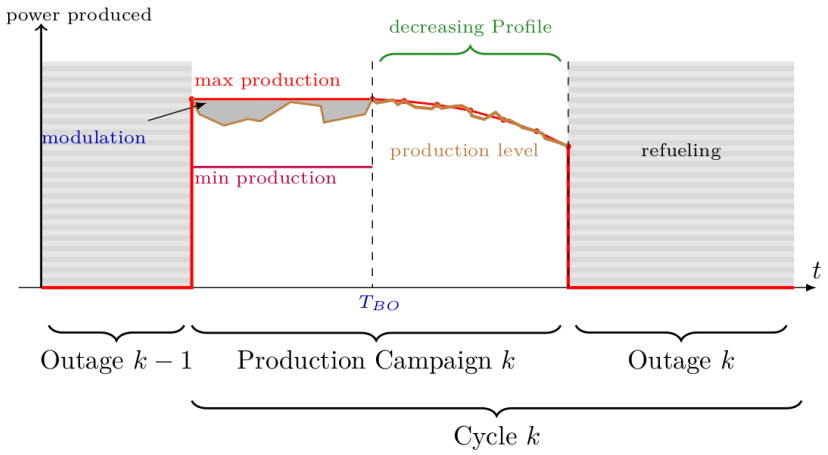

Figure 1: Illustration of the specific constraints related to nuclear units.

with cost matching expected selling prices on electricity markets and negative production is added to the non-nuclear fleet to prevent any overcapacity with respect to the demand.

2.2. Nuclear unit production constraints

A nuclear unit operates in a cyclic but non-periodic way. Every *cycle* starts with a production campaign which might be divided into two phases. During the first phase, the unit may be operated at intermediate power in order to save fuel for later on. The total fuel saving, called *modulation*, is limited for each cycle. If the fuel level reaches a given threshold, called *BO* in the following, the unit operation enters its second phase where production has to follow a non-linear *decreasing profile* starting from full power and decreasing each week from about 3% until the next outage starts. Figure 1 gives an illustration of a nuclear cycle.

A cycle ends with an outage during which maintenance is performed and a given amount of nuclear fuel is reloaded. In practice, a fraction (a third or quarter) of the assemblies in the core of the reactor are replaced by fresh ones. Deciding to start an outage period during the first phase, i.e. with a fuel level greater than the BO level, is called *a stop by anticipation*; it can occur only provided the fuel level is below a given fuel level called *maximum anticipation*. Conversely, a stop during the second phase is called a *stop in prolongation*. Note that, from the fuel perspective, a stop in prolongation is economically more interesting than a stop by anticipation, as a prolongation leads to a better use of the removed fuel assemblies. However, the unit must be stopped at the latest once the fuel level has reached a threshold called *maximum prolongation*.

2.3. Outage scheduling constraints

Nuclear units refueling and maintenance operations are complex industrial tasks that require personnel with specific and possibly rare skills, and dedicated equipment. Hence, the number of outages sharing a specific resource is limited

at any time. There is a large variety of such resource constraints, but the most widely used ones can be classified in one of the following two categories:

- Local power plant constraints: nuclear units are located on 19 power plants. All but one power plant comprise either two or four units. Each of such power plants has the required personnel to perform at most one outage at a time. In addition, the outages of any two units at the same power plant should be separated by several weeks. As for the six unit power plant, the outages of at most two units can be performed at a time.
- Global power plant constraints: when an outage duration is less than a month, only nuclear fuel can be reloaded. When an outage duration is more than a month, maintenance tasks are also performed and a very specific equipment is required. Hence, the number of simultaneous outages with a duration more than a month is limited for the whole nuclear fleet.

Nomenclature and notations

In this article, sets, vectors and matrices are indicated in boldface, whereas a scalar is in non-boldface.

Indices

$t \in \mathbf{T}$	index of weeks
$j \in \mathbf{J}$	index of non-nuclear plants
$i \in \mathbf{I}$	index of nuclear units
$k \in \mathbf{K_i}$	cycle index of nuclear unit i
$c \in \mathbf{R}$	scheduling constraint index
$\omega \in \mathbf{\Omega}$	index of scenarios

Input Data

D_t^ω total demand for week t in scenario ω

For each nuclear unit i:

S_i	initial fuel level
$\overline{P}_{it}$	maximum production during week t

For each nuclear unit i and each cycle k:

A_{ik}	maximum fuel level at the beginning of the outage
M_{ik}	maximum modulation during the production period
DP_{ik}	minimum fuel level at the beginning of the outage
ED_{ik}	early date of the outage
DD_{ik}	due date of the outage

For each non-nuclear unit j:

$\overline{P}_{jt}^{\omega}$	maximum production during week t in scenario ω
C_{jt}^{ω}	leasing cost during week t in scenario ω

For each scheduling constraint r:

N_{rt}	maximum number of resources available during week t
$\boldsymbol{I}_r$	time interval (in weeks) during which constraint r is active
$\boldsymbol{O}_r$	set of outages defined by pair (i,k) involved in constraint r

$L_r(i,k)$ delay (in weeks) after the beginning of outage $(i,k) \in \boldsymbol{O}_r$ for the consumption of one resource for each nuclear unit i and cycle k

$D_r(i,k)$ duration of resource consumption for outage $(i,k) \in \boldsymbol{O}_r$

Energy is given in *Equivalent Full Power*, denoted by *EFP*, corresponding to the energy produced in one week at full power for the corresponding nuclear unit.

3. Extended formulations

In this section, the nuclear constraints of each unit are captured in a dedicated graph, thus leading to a *network flow formulation*. Such a formulation involves additional variables w.r.t. original variables, e.g., outage dates, thus leading to a so called *extended formulation*. Indeed an extended formulation of the original polyhedron is a polyhedron whose projection onto the original variables is the original polyhedron.

3.1. Assumptions

Assumption 1. *Similarly to [17] and [11], we assume a discretization of the fuel levels that can be reached at the end of a production cycle.*

Assumption 2. *A production campaign with modulation greater than a week of production will contain a modulation corresponding to an integer number of weeks during which the unit is operated at minimum power instead of maximum power. It implies in particular that BO level will be reached at the end of a week, allowing a decreasing profile period to start at the beginning of the next week.*

Assumption 3. *A production campaign that involves a significant modulation, i.e more than one EFP, cannot be stopped before the fuel level reaches BO level.*

Assumption 4. *Inversely, a production campaign can be stopped by anticipation provided it involves no modulation, except the minimum required to end the cycle with the nearest discrete fuel level available. Note that the corresponding modulation must be lower than one EFP (Equivalent Full Power).*

Assumption 5. *The fuel level at the end of a given production period and the starting time of each decreasing profile are common to all scenarios. It implies in particular that the total energy saved by modulation is common for all scenarios as well.*

Assumption 1 is a mild assumption used to convert continuous fuel levels in discrete states decisions. Assumptions 2-4 are slightly stronger assumptions used to define production arcs corresponding to production decisions which translate into bounds on production variables in Section 3.3. Assumption 2 separates decreasing profile periods from production periods. Assumptions 3 and 4 are standard from an economical point of view. The rationale behind is that using modulation before a stop with anticipation would increase the amount of fuel loss during an outage. Finally Assumption 5 is the strongest assumption used to separate first stage decisions, i.e., outages dates and fuel levels, from second stage production dispatching in each scenario. As for Assumption 1, it can be argued that matter, while production dispatching results from the former decisions.

Note that the cost and feasibility w.r.t. the demand of a given scenario can be evaluated once production is dispatched. Assumption 5 is used to obtain an efficient formulation in Section 3.5 and in decomposition schemes presented in Section 4. Under these assumptions, we assume that there exists at least one feasible production plan for each nuclear unit.

3.2. Transition graph

Thanks to the previous assumptions, we can associate to each nuclear unit i a graph $\boldsymbol{G}_i = (\boldsymbol{V}_i, \boldsymbol{E}_i)$. Each path from the source to the sink of this graph corresponds to a plan satisfying the following constraints for unit i: time window for each outage, maximum fuel level for each refueling both minimum and maximum fuel level, and maximum modulation for each cycle.

Each vertex is uniquely associated with a flag f, a cycle index k, a fuel level index a (Assumption 1) and a week t, while each arc corresponds to either an outage or a production period. Vertices are partitioned in four classes:

$(f = \textbf{Begin}, k, a, t)$ called *Begin vertex* corresponds to the start of the k^{th} production period taking place at the beginning of week t, while a corresponds to the index of the fuel level at the end of the preceding production period.

$(f = \textbf{End}, k, a \geq 1, t)$ called *Anticipation vertex* corresponds to the end of the k^{th} production period taking place at the end of week $t-1$ with a positive fuel level corresponding to a weeks at full power.

$(f = \textbf{BO}, k, a = 0, t)$ called *BO vertex* corresponds to the case where the BO level is reached in the k^{th} cycle at the end of week $t-1$. An outage or a decreasing profile period may start at the beginning of week t.

$(f = \textbf{End}, k, a < 0, t)$ called *DP vertex* corresponds to the end of the k^{th} production period taking place at the beginning of week t following a decreasing profile of $-a$ weeks.

Note that the information defining vertices includes the exact fuel level of the unit in week t. The source node of $\boldsymbol{G}_i$ is a fictitious node whose data are adjusted in order to reflect the initial condition of the unit at the outset of the planning horizon.

An arc is defined by a starting week t_1 and an ending week $t_2 - 1$. Assumptions 2 to 4 let us use only arcs in the following four categories:

Full-power arcs linking a *Begin vertex* to an *Anticipation vertex*. Each arc is defined only if t_2 is within the time window for the following outage and if the targeted fuel level is below the maximum anticipation of the current cycle.

Modulation arcs linking a *Begin vertex* to a *BO vertex*. Such arcs correspond to a fuel reload and a production leading to the *BO* level at the end of week $t_2 - 1$. It is defined if and only if the amount of modulation necessary to reach BO level at the end of week $t_2 - 1$ is below the maximum modulation value of the current cycle.

Decreasing profile arcs linking a *BO vertex* to a *DP vertex*. Such arcs correspond to the decreasing profile of the current cycle from week t_1 to $t_2 - 1$. Week t_2 has to be within the time window of the following outage.

Outage arcs linking a BO or End vertex to a Begin vertex relative to the next cycle. Such arcs represent an outage between weeks t_1 and $t_2 - 1$ with a fuel reload. Note that the length of the current cycle's outage should equal $t_2 - t_1 - 1$.

The sink of the graph is a fictitious vertex linked to all vertices going beyond the horizon t by a fifth kind of arc called *arc to sink*.

Let us consider an instance with a single unit as an illustrative example on the graph construction. For ease of presentation, the index of the unit is omitted. By convention, the initial cycle index and initial week are 0. The entries given in terms of EFP for the first cycle $k = 0$ are the following : $S = 9.8$, $A_0 = 2.5$, $M_0 = 3$, $DP_0 = 8$, while the early and due dates are $ED_0 = 7$ and $DD_0 = 11$. The first vertex with a positive fuel level is a Begin vertex. Hence, the source vertex s of the graph is with label (Begin,0,0,0). For full-power arcs, the unit must produce at full power at least for eight weeks for the fuel level initially at $S = 9.8$ to be less than $A_0 = 2.5$. The discretized final fuel levels correspond to integers in terms of EFP, hence every production period will at least contain a modulation of 0.2 EFP. At the beginning of week 8, the fuel level is 2 and the production period can end as $ED_0 \leq 8 \leq DD_0$. A full-power arc is added to the graph leading to vertex (End,0,2,8). Similarly, another full-power arc leads to vertex (End,0,1,9), which corresponds to produce during one additional week. No more full power arcs can be added. Reaching BO level through a modulation arc at the beginning of week 10 is possible with a modulation of 0.2 EFP between week 0 and 9, the corresponding vertex is (BO,0,0,10). With a modulation of 1.2 EFP between week 0 and 9, another vertex (BO,0,0,11) can be added. As the outage due date has been reached, no more vertices can be added, even though it would be possible w.r.t. M_0 to increase modulation. Decreasing profile periods correspond to keep producing below BO level, i.e., with a negative fuel level. The time window allows producing until the end of week 10, i.e., beginning of week 11. Hence a decreasing profile arc is created from vertex (BO,0,0,10) to a new vertex (End,0,-1,11). As the outage due date has been reached, no more vertices can be added, even though it would be possible w.r.t. DP_0 to keep producing along the decreasing profile. As for outage arcs, an outage can begin at each End and BO vertices. Figure 2 shows the resulting graph at the beginning of cycle 1.

In the remainder of the article, each arc is associated with an index e where $e = (u, v)$ represents the arc with origin vertex u and destination vertex v.

3.3. Additional precomputed data

In order to write our models, we introduce additional parameters computed from the input data.

First, we associate with each arc bounds that specify how energy production is constrained when it is chosen in a solution. Given an arc $e \in \boldsymbol{E}_i$, starting at the beginning of week $t_1(e)$ and ending at the end of the week $t_2(e) - 1$, we define $\bar{p}_e(t)$ (resp. $\underline{p}_e(t)$), $t \in \{t_1(e), \ldots, t_2(e) - 1\}$ such that $\sum_{t'=t_1}^{t} \bar{p}_e(t')$ (resp. $\sum_{t'=t_1}^{t} \underline{p}_e(t')$) is the maximum (resp. minimum) possible energy produced by plant i between weeks $t_1(e)$ and t, taking into account the type of arc e and its

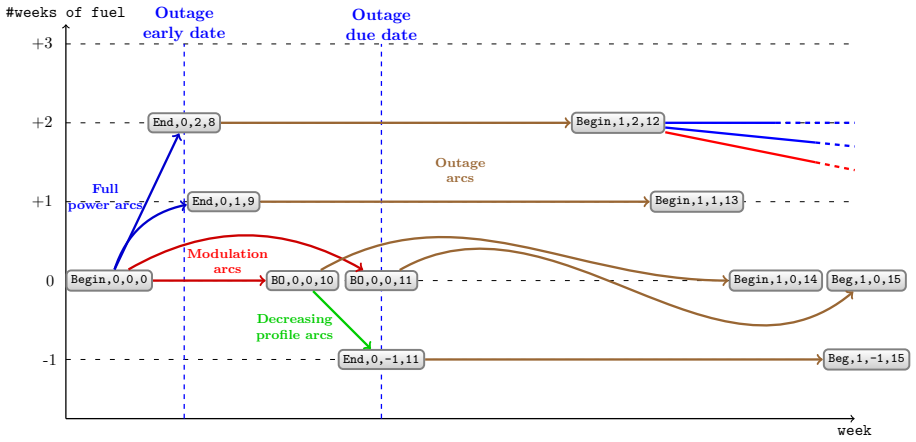

Figure 2: Graph relative to a single unit at the end of the first cycle.

prescribed start and end fuel levels. For *full power* and *decreasing profile arcs*, $\bar{p}_e(t) = \underline{p}_e(t)$ for all t. Note that the use of those bounds allows modeling non-linear decreasing profile constraint through linear constraints involving integer arc variables. For *outage arcs*, $\bar{p}_e(t) = \underline{p}_e(t) = 0$ for all t. For a *modulation arc*, $\bar{p}_e(t)$ (resp. $\underline{p}_e(t)$), $t \in \{t_1(e), \ldots, t_2(e) - 1\}$, is computed by placing the maximum amount of production as soon as (resp. as late as) possible in the corresponding cycle (see Figure 3). Note that those bounds prescribe a precise cumulative energy produced over the time period spanned by arc e. By convention, we set $\bar{p}_e(t) = \underline{p}_e(t) = 0$ for all e, w such that $e, w \notin [t_1(e), t_2(e) - 1]$.

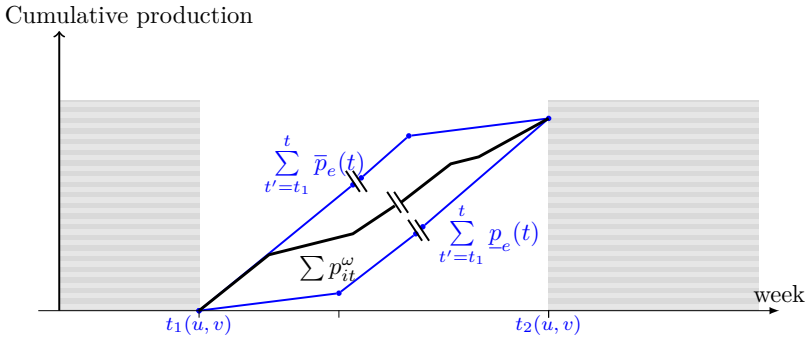

Figure 3: Bounds on cumulative energy produced for a given arc (u, v).

Second, we aim at writing a compact expression for the set $\boldsymbol{R}$ of various scheduling constraints. Each constraint $r \in \boldsymbol{R}$ involves a set $\boldsymbol{O}_r$ of outages each identified with plant i and the cycle k. When outage (i, k) starts at week $t \in \boldsymbol{I}_r$, one unit of resource r is used from week $t + L_r(i, k)$ to week $t + L_r(i, k) + D_r(i, k) - 1$. Based on those data, we derive the set of *outage arcs*

that are involved in each constraint r at week t:
$\boldsymbol{O}_{rt} = \{e \text{ outage arc} : t_1(e) \leq t - L_r(i,k) \leq t_1(e) + D_r(i,k), e \in \boldsymbol{E}_i, (i,k) \in \boldsymbol{O}_r\}$.

3.4. Arc flow-based mixed integer programming formulation

We now introduce a natural formulation in the sense it is based on natural variables arising from the transition graph defined in Section 3.2 and the precomputed data in Section 3.3. This is a preliminary step before introducing additional variables leading to more efficient formulations.

For nuclear unit i and each arc e of graph $\boldsymbol{G}_i = (\boldsymbol{V}_i, \boldsymbol{E}_i)$, let δ_e be a 0-1 variable which takes on a value of 1 if the arc is used to define the production plan for nuclear unit i, 0 otherwise. We note respectively $s(\boldsymbol{G}_i)$ and $t(\boldsymbol{G}_i)$ the source and the sink of $\boldsymbol{G}_i$. For each scenario ω, let p^{ω}_{iw} be a continuous variable defining the energy produced by nuclear unit i during week w. Similarly let p^{ω}_{jw} be a continuous variable defining the energy produced by non-nuclear nuclear unit j for week w under scenario ω. An intermediary variable $\underline{q}_{it}$ (resp. $\overline{q}_{it}$) is introduced to set the lower bound (resp. upper bound) on the total energy produced by nuclear unit i up to time period t in any scenario ω, i.e., such energy is bound to lie in the prescribed envelope defined in Section 3.3. The resulting formulation for NOPP with $(\boldsymbol{\delta}, \boldsymbol{p})$ as mandatory variables and $(\underline{\boldsymbol{q}}, \overline{\boldsymbol{q}})$ as intermediate variables is denoted by $F_{(\underline{\boldsymbol{q}}, \overline{\boldsymbol{q}})}(\boldsymbol{\delta}, \boldsymbol{p})$ and is as follows.

$$\min \frac{1}{\boldsymbol{\Omega}} \sum_{j,t,\omega} C^{\omega}_{jt} p^{\omega}_{jt} \tag{1}$$

$$\sum_{(u,v) \in \boldsymbol{O}_{rt}} \delta_{(u,v)} \leq N_{rt} \qquad \forall r \in \boldsymbol{R}, t \in \boldsymbol{I}_r \tag{2}$$

$$\sum_{(u,v) \in \boldsymbol{E}_i} \delta_{(u,v)} - \sum_{(v,u) \in \boldsymbol{E}_i} \delta_{(v,u)} = 0 \quad \forall i, u \in \boldsymbol{V}_i \backslash \{s(\boldsymbol{G}_i), t(\boldsymbol{G}_i)\} \tag{3}$$

$$\sum_{(s(\boldsymbol{G}_i),v) \in \boldsymbol{E}_i} \delta_{(s(\boldsymbol{G}_i),v)} = 1 \qquad \forall i \tag{4}$$

$$\delta_{(u,v)} \in \{0,1\} \qquad \forall i, (u,v) \in \boldsymbol{E}_i \tag{5}$$

$$\underline{q}_{it} = \sum_{(u,v) \in \boldsymbol{E}_i} \underline{p}_{(u,v)}(t) \delta_{(u,v)} \qquad \forall i, t \tag{6}$$

$$\overline{q}_{it} = \sum_{(u,v) \in \boldsymbol{E}_i} \overline{p}_{(u,v)}(t) \delta_{(u,v)} \qquad \forall i, t \tag{7}$$

$$\sum_{t'=0}^{t} \underline{q}_{it'} \leq \sum_{t'=0}^{t} p^{\omega}_{it'} \leq \sum_{t'=0}^{t} \overline{q}_{it'} \qquad \forall i, t, \omega \tag{8}$$

$$\sum_{i} p^{\omega}_{it} + \sum_{j} p^{\omega}_{jt} = D^{\omega}_{t} \qquad \forall t, \omega \tag{9}$$

$$0 \leq p^{\omega}_{it} \leq \overline{P}_{it} \qquad \forall i, t, \omega \tag{10}$$

$$0 \leq p^{\omega}_{jt} \leq \overline{\mathrm{P}}^{\omega}_{jt} \qquad \forall j, t, \omega \tag{11}$$

Objective function (1) minimizes the average total production cost of non-nuclear

units over all considered scenarios ω, (2) corresponds to the scheduling constraints with limited resource, (4) are flow constraints imposing that one feasible plan should be assigned to each nuclear unit. Then (6) and (7) define bounds on energy production corresponding to arc flow decisions and enforced to nuclear production by (8) for each week w and each scenario ω. Finally (9) ensures that the demand is satisfied each week t under each scenario ω.

3.5. Dedicated arc flow-based MIP formulation

The arc flow formulation presented in Section 3.4 is quite straightforward bearing in mind that total energy to be produced by each nuclear unit is bound to lie in a given envelope as defined in Section 3.3. However, constraints (6)-(8) induce triangular dense structures linking $\boldsymbol{\delta}$ variables to nuclear production variables for each scenario ω. Such structure is likely to impair the performance of the formulation especially anticipating to solve large-scale NOPP instances.

In this section, the key idea is to take advantage more explicitly of the definition of arcs combined with Assumption 5, which states that the fuel level at the end of any production period is the same for all scenarios. Formally, for a given production arc $e \in \boldsymbol{E}_i$:

$$\sum_{t=t_1(e)}^{t_2(e)} \underline{p}_{it} = \sum_{t=t_1(e)}^{t_2(e)} \overline{p}_{it} = \sum_{t=t_1(e)}^{t_2(e)} p_{it}^{\omega} \quad ,\forall \omega \tag{12}$$

Let us introduce, for each nuclear unit i and week t, set $\boldsymbol{A}_{it} \subset \boldsymbol{E}_i$, the set of *active arcs* at time t, i.e., $\boldsymbol{A}_{it} = \{e \in \boldsymbol{E}_i | t_1(e) \leq t < t_2(e)\}$. Interested reader may check that the following proposition follows from the left equality in (12) combined with the definition of $\underline{\boldsymbol{p}}$ and $\overline{\boldsymbol{p}}$:

Proposition 1.

$$\sum_{t'=0}^{t} \sum_{e \in \boldsymbol{E}_i} \left(\overline{p}_e(t') - \underline{p}_e(t')\right) \delta_e = \sum_{e \in \boldsymbol{A}_{it}} \sum_{t'=t_1(e)}^{t} \left(\overline{p}_e(t') - \underline{p}_e(t')\right) \delta_e$$

Then we can rewrite equations (6)-(8) without q-variables for a given unit i, a time period t and a scenario ω:

$$\begin{aligned}
&\sum_{t'=0}^{t} \sum_{e \in \boldsymbol{E}_i} \underline{p}_e(t')\delta_e \leq \sum_{t'=0}^{t} p_{it'}^{\omega} \leq \sum_{t'=0}^{t} \sum_{e \in \boldsymbol{E}_i} \overline{p}_e(t')\delta_e \\
\Leftrightarrow &\sum_{t'=0}^{t} \sum_{e \in \boldsymbol{E}_i} \left(\overline{p}_e(t') - \underline{p}_e(t')\right) \delta_e \geq \sum_{t'=0}^{t} \left(\sum_{e \in \boldsymbol{E}_i} \overline{p}_e(t')\delta_e - p_{it'}^{\omega} \right) \geq 0 \\
\Leftrightarrow &\sum_{e \in \boldsymbol{A}_{it}} \sum_{t'=t_1(e)}^{t} \left(\overline{p}_e(t') - \underline{p}_e(t')\right) \delta_e \geq \sum_{t'=0}^{t} \left(\sum_{e \in \boldsymbol{E}_i} \overline{p}_e(t')\delta_e - p_{it'}^{\omega} \right) \geq 0
\end{aligned} \tag{13}$$

Note that only active arcs in $\boldsymbol{A}_{it}$ have a non-zero contribution in the left hand side of Equation (13). Moreover the contribution of the decreasing profile

and outage arcs in this left hand side is zero; the same holds for full power arcs, except in the case of arcs with marginal modulation for which the contribution stays close to zero (see Assumption 4).

For the right hand side of Equation (13) we need to use a difference equation in order to keep only contributions of active arcs. Let us introduce a new continuous variable s^{ω}_{it} defined as the difference between the upper bound on the total energy produced and the real production by nuclear unit i up to time period t in scenario ω. Then, we can represent the evolution of this variable by a difference equation involving active arcs $\boldsymbol{A}_{it}$ and production variable p^{ω}_{it}:

$$\begin{aligned} s^{\omega}_{it} &= \sum_{t'=0}^{t} \left(\sum_{e \in \boldsymbol{E}_i} \overline{p}_e(t)\delta_e - p^{\omega}_{it'} \right) \\ \Leftrightarrow s^{\omega}_{it} - s^{\omega}_{i,t-1} &= \sum_{e \in \boldsymbol{A}_{it}} \overline{p}_e(t)\delta_e - p^{\omega}_{it} \\ \Leftrightarrow s^{\omega}_{it} + p^{\omega}_{it} &= s^{\omega}_{i,t-1} + \sum_{e \in \boldsymbol{A}_{it}} \overline{p}_e(t)\delta_e \end{aligned} \tag{14}$$

Finally we can rewrite Equation (13) using variables s as:

$$\forall i, t, \omega \qquad \sum_{e \in \boldsymbol{A}_{it}} \sum_{t'=t_1(e)}^{t} \left(\underline{p}_e(t') - \overline{p}_e(t') \right) \delta_e \geq s^{\omega}_{it} \geq 0 \tag{15}$$

The following equivalent MIP is thus derived:

$$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p}) : \min \left\{ \frac{1}{\boldsymbol{\Omega}} \sum_{j,t,\omega} C^{\omega}_{jt} p^{\omega}_{jt} : (2)-(5), (14)-(15), (9)-(11) \right\}$$

From preliminary experiments (see Section 5), formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ appears to outperform by far formulation $F_{(\underline{\boldsymbol{q}},\overline{\boldsymbol{q}})}(\boldsymbol{\delta}, \boldsymbol{p})$ introduced in Section 3.4. The rationale behind introducing formulation $F_{(\underline{\boldsymbol{q}},\overline{\boldsymbol{q}})}(\boldsymbol{\delta}, \boldsymbol{p})$ is twofold. First it helps deriving formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$, which is in comparison more involved. Second it provides a reference to assess the improved performance. For the rest of the article, formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ is retained and formulation $F_{(\underline{\boldsymbol{q}},\overline{\boldsymbol{q}})}(\boldsymbol{\delta}, \boldsymbol{p})$ discarded.

3.6. Splitting first and second stages through capacity variables

In the vein of the reformulation for problems with a fixed technology matrix exposed in [3], Chap. 3, Section 1, we project the first-stage variables $\boldsymbol{\delta}$ onto the capacity of production or modulation relative to the plant schedules corresponding to $\boldsymbol{\delta}$. Note that such a capacity is the only relevant information for the second stage problem. This technique is of special interest in the context of a double decomposition, where reformulations lead to an exponential (in terms of the size of input data) number of variables involved in an exponential number of constraints. The general solving process we design is based on the dynamic generation of the model. That implies heavy computational burden when adding each constraint or column, since the number of new coefficients to set at this occasion is potentially huge. The step of the algorithm corresponding to setting those additional coefficients will be referred to as *projection*. Moreover, the

overall number of non-zero coefficients in the MIP formulation is very large as well. We investigated two ways of projecting the first-stage variables $\boldsymbol{\delta}$ onto the second-stage constraints, leading to equivalent formulations but with different computational benefits.

The first option is to define first-stage variables $\overline{q}_{it}$, $i \in \mathbf{I}$, $t \in \mathbf{T}$, which are equal to the maximum possible total energy produced by i up to week t, restricted to the cycle at t, and m_{it} the difference between the upper and lower bounds of the production envelop and hence the upper bound on variable s^{ω}_{it}:

$$\sum_{e \in \boldsymbol{E}_i} \overline{p}_e(t).\delta_e = \overline{q}_{it} \qquad \forall i, t \tag{16}$$

$$\sum_{e \in \boldsymbol{A}_{it}} \sum_{t'=t_1(e)}^{t} (\overline{p}_e(t') - \underline{p}_e(t')).\delta_e = m_{it} \qquad \forall i, t \tag{17}$$

$$\overline{q}_{it} + s^{\omega}_{i,t-1} = s^{\omega}_{i,t} + p^{\omega}_{it} \qquad \forall i, t, \omega \tag{18}$$

$$s^{\omega}_{it} \leq m_{it} \qquad \forall i, t, \omega \tag{19}$$

The second option is to use cumulative capacity variables $\overline{cq}_{it}$ (resp. $\underline{cq}_{it}$), $i \in \mathbf{I}$, $t \in \mathbf{T}$, defined as the maximum (resp. minimum) total energy produced by unit i up to week t:

$$\sum_{e \in \boldsymbol{E}_i} \sum_{t'=t_1(e)}^{t} \underline{p}_e(t').\delta_e = \underline{cq}_{it} \qquad \forall i, t \tag{20}$$

$$\sum_{e \in \boldsymbol{E}_i} \sum_{t'=t_1(e)}^{t} \overline{p}_e(t').\delta_e = \overline{cq}_{it} \qquad \forall i, t \tag{21}$$

$$\overline{cq}_{it} - \overline{cq}_{i,t-1} + s^{\omega}_{i,t-1} = s^{\omega}_{i,t} + p^{\omega}_{it} \qquad \forall i, t, \omega \tag{22}$$

$$s^{\omega}_{it} \leq (\overline{cq}_{it} - \underline{cq}_{it}) \qquad \forall i, t, \omega \tag{23}$$

The corresponding resulting formulations are the following:

$$F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p}) : \min \left\{ \frac{1}{\boldsymbol{\Omega}} \sum_{j,t,\omega} C^{\omega}_{jt} p^{\omega}_{jt} : (2) - (5), (16) - (18), (9) - (11) \right\}$$

$$F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p}) : \min \left\{ \frac{1}{\boldsymbol{\Omega}} \sum_{j,t,\omega} C^{\omega}_{jt} p^{\omega}_{jt} : (2) - (5), (20) - (22), (9) - (11) \right\}$$

4. Solution approaches

This section describes the proposed solution algorithms for NOPP. Such algorithms are based on the mathematical programming formulations presented in Section 3.5 and 3.6. As the intent is to solve large-scale NOPP instances, a double decomposition approach is considered, thus leading to a so-called *reformulated problem*. The principle is to solve iteratively small subproblems,

which prevents us from solving too large of a problem. To ease notation, the different formulations are cast into a generic linear matrix formulation. The idea is to make it possible to present the Dantzig-Wolfe and Benders decomposition schemes in a general setting suited for all formulations. The row-and-column generation algorithm optimizing the linear relaxation of the reformulated problem is then presented. Finally the way to get near-optimal feasible solutions for NOPP from this relaxation is explained.

4.1. Generic formulation

The proposed generic formulation, called F_{Gen}, is introduced to cast the mathematical program corresponding to all formulations in a general setting. Such generic formulation involves vectors to put together subsets of decision variables, and matrices to capture the structure of all formulations.

All formulations presented in Section 3.5 feature a planning for each nuclear plant as a common structure. Such planning is defined in constraints (3)-(5), which can be rewritten as (25)-(26) in F_{Gen}. More precisely matrix $\boldsymbol{\Delta}$ and vector $\boldsymbol{d}$ are used to rewrite the shortest path constraints (3)-(4). For a given scenario $\omega \in \boldsymbol{\Omega}$, nuclear and non-nuclear power production variables, $(p^{\omega}_{it})_{i\in\mathbf{I},\mathbf{t}\in\mathbf{T}}$ and $(p^{\omega}_{jt})_{j\in\mathbf{J},\mathbf{t}\in\mathbf{T}}$, respectively, are included into a single vector $\boldsymbol{p}^{\omega}$, so that demand constraints (9) and bounds on production (10)-(11) are cast as (30) in F_{Gen}. Cost vectors $\bar{\boldsymbol{C}}^{\omega}$, $\omega \in \boldsymbol{\Omega}$, showing in the objective (24), take value $\bar{C}^{\omega}_{it} = 0$ for $i \in \mathbf{I}$ and $t \in \mathbf{T}$, and $\bar{C}^{\omega}_{jt} = \frac{1}{|\boldsymbol{\Omega}|} C^{\omega}_{jt}$ for $j \in \mathbf{J}$ and $t \in \mathbf{T}$.

The proposed three formulations differ from one another in the way they link the plant plannings to the actual corresponding power production. We introduce a vector of abstract variables $\boldsymbol{\xi}$ in the sense they replace either variables $\boldsymbol{s}$ from formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ or $[\overline{\boldsymbol{q}}, \boldsymbol{s}]$ (resp. $[\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}}, \boldsymbol{s}]$) from $F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ (resp. $F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$) and their dimension changes accordingly. Matrices $\mathbf{A}_0$, $\boldsymbol{\Xi}_0$ and $\boldsymbol{\Theta}^{\omega}_0$ and vector $\boldsymbol{b}^{\omega}_0$, $\omega \in \boldsymbol{\Omega}$, are used to rewrite constraints (14)-(15) from $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ as constraints (27) in F_{Gen}. They are with zero dimension in the case of alternative formulations, namely $F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ and $F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$. Matrices $\mathbf{A}_1$ and $\boldsymbol{\Xi}_1$ and vector $\boldsymbol{b}_1$ are used in constraints (28) to rewrite constraints (2) relative to resource-constrained scheduling along with complementary constraints induced by abstract variables $\boldsymbol{\xi}$, namely constraints (16)-(17) (resp. (20)-(21)) in formulation $F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ (resp. $F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$). Matrices $\boldsymbol{\Xi}_2$ and $\boldsymbol{\Theta}^{\omega}_2$ and vector $\boldsymbol{b}^{\omega}_2$, $\omega \in \boldsymbol{Omega}$ are used in constraints (29) to rewrite constraints (18)-(19) (resp. (22)-(23)) linking power production $\boldsymbol{p}^{\omega}$ to abstract variables $\boldsymbol{\xi}$ in formulation $F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ (resp. $F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$). They are with zero dimension in the case of formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$. Note that generic constraints (28) involve only first-stage variables, while (27) and (29) involve first- and second-stage variables. The correspondence between the abstract variables and the generic constraints in F_{Gen} and all formulations is summarized in Table 1.

$$F_{Gen} : \min \quad \sum_{\omega} \bar{\boldsymbol{C}}^{\omega\top} \boldsymbol{p}^{\omega} \tag{24}$$

$$s.t. \quad \boldsymbol{\Delta}\boldsymbol{\delta} = \boldsymbol{d} \tag{25}$$

$$\boldsymbol{\delta} \in \{0, 1\} \tag{26}$$

$$\mathbf{A}_0\boldsymbol{\delta} + \boldsymbol{\Xi}_0\boldsymbol{\xi} + \boldsymbol{\Theta}_0^\omega \boldsymbol{p}^\omega \geq \boldsymbol{b}_0^\omega \quad \forall\omega \tag{27}$$
$$\mathbf{A}_1\boldsymbol{\delta} + \boldsymbol{\Xi}_1\boldsymbol{\xi} \geq \boldsymbol{b}_1 \tag{28}$$
$$\boldsymbol{\Xi}_2\boldsymbol{\xi} + \boldsymbol{\Theta}_2^\omega \boldsymbol{p}^\omega \geq \boldsymbol{b}_2^\omega \quad \forall\omega \tag{29}$$
$$\mathbf{P}^\omega \boldsymbol{p}^\omega = \boldsymbol{D}^\omega \quad \forall\omega \tag{30}$$
$$\boldsymbol{p}^\omega \geq \mathbf{0} \quad \forall\omega \tag{31}$$
$$\boldsymbol{\xi} \geq \mathbf{0} \tag{32}$$

F_{Gen}	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	$F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	$F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$
$\boldsymbol{\xi}$	$\boldsymbol{s}$	$[\overline{\boldsymbol{q}}, \boldsymbol{m}]$	$[\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}}, \boldsymbol{s}]$
(27)	(14), (15)	-	-
(28)	(2)	(2),(16),(17)	(2), (20),(21)
(29)	-	(18),(19)	(22),(23)

Table 1: Correspondence between the generic formulation F_{Gen} and all proposed formulations $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$, $F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ and $F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$. Mark "-" indicates formulations for which the generic constraint (27) (resp. (29)) vanishes, i.e., $F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ and $F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ (resp. $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$).

4.2. Dantzig-Wolfe reformulation

Constraints (25)-(28) present a classical structure which is suitable for column generation. As constraints (25)-(26) are independent for each power plant i, they define a subproblem with binding constraints (27)-(28).

Let $\mathcal{G}_i$ be the set of feasible paths satisfying constraints (25) for nuclear unit i. Let λ_{ig} be the decision variable associated with choosing path $g \in \mathcal{G}_i$ for nuclear unit i and $\boldsymbol{\Lambda}^g \in \{0,1\}^{|\boldsymbol{E}_i|}$ the binary vector such that $\Lambda_e^g = 1$ if and only if arc $e \in \boldsymbol{E}_i$ is part of path g. For $i \in \mathbf{I}$ and $e \in \boldsymbol{E}_i$, we can now write $\delta_e = \sum_{g\in\mathcal{G}_i} \Lambda_e^g \lambda_{ig}$, *i.e.*, in matrix form $\boldsymbol{\delta} = \boldsymbol{\Lambda}\boldsymbol{\lambda}$. Moreover, let $\mathbf{H} \in \{0,1\}^{|\mathbf{I}|\times\sum_{\mathbf{i}}|\mathcal{G}_\mathbf{i}|}$, such that $H_i^g = 1$ if and only if $g \in \mathcal{G}_i$. This leads to the following Dantzig-Wolfe reformulation of F_{Gen}:

$$F_{Gen}^{DW}: \min \quad \sum_\omega \bar{\boldsymbol{C}}^{\omega\top} \boldsymbol{p}^\omega \tag{33}$$
$$s.t. \quad \mathbf{H}\boldsymbol{\lambda} = \mathbf{1} \tag{34}$$
$$\mathbf{A}_0\boldsymbol{\Lambda}\boldsymbol{\lambda} + \boldsymbol{\Xi}_0\boldsymbol{\xi} + \boldsymbol{\Theta}_0^\omega \boldsymbol{p}^\omega \geq \boldsymbol{b}_0^\omega \quad \forall\omega \tag{35}$$
$$\mathbf{A}_1\boldsymbol{\Lambda}\boldsymbol{\lambda} + \boldsymbol{\Xi}_1\boldsymbol{\xi} \geq \boldsymbol{b}_1 \tag{36}$$
$$(29), (30), (31), (32)$$
$$\boldsymbol{\lambda} \in \{0,1\}^{\sum_i |\mathcal{G}_i|} \tag{37}$$

Exponentially-many $\boldsymbol{\lambda}$-variables are involved in constraints (35), which are replicated for each scenario ω. Recall constraints (35) is a reformulation of constraints (27) actually used only in formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$. Interestingly the

use of additional variables in formulations $F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta},\boldsymbol{s},\boldsymbol{p})$ and $F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta},\boldsymbol{s},\boldsymbol{p})$ allows for elimination of $\boldsymbol{\lambda}$-variables from second-stage constraints. The benefit is even clearer when the second-stage problem is reformulated with an exponential number of constraints, as shown in the next section. The Dantzig-Wolfe reformulation of $F(\boldsymbol{\delta},\boldsymbol{s},\boldsymbol{p})$ (resp. $F_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\delta},\boldsymbol{s},\boldsymbol{p})$ or $F_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\delta},\boldsymbol{s},\boldsymbol{p})$) is denoted by $F^{DW}(\boldsymbol{\lambda},\boldsymbol{s},\boldsymbol{p})$ (resp. $F^{DW}_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\boldsymbol{\lambda},\boldsymbol{s},\boldsymbol{p})$ or $F^{DW}_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\lambda},\boldsymbol{s},\boldsymbol{p})$).

4.3. Benders reformulation

We propose to cope with the second-stage part of the problem using Benders reformulation [2]. To avoid dealing with both optimality and feasibility cuts, we first move the second-stage objective value into constraints. Hence while only feasibility cuts are written, some of them can be interpreted as optimality cuts. The feasibility cuts are derived from an appropriate feasibility subproblem (see *e.g.* [19]). We use the *multi-cut* approach, which is to deal with feasibility and optimality conditions for each scenario independently.

Given the first-stage solution $(\boldsymbol{\lambda},\boldsymbol{\xi})$, let us introduce the recourse function $R^{\omega}(\boldsymbol{\lambda},\boldsymbol{\xi})$, $\omega \in \boldsymbol{\Omega}$, equal to the optimal cost of the second-stage solution in scenario ω if one exists, or equal to ∞ if the second stage problem is infeasible. We also use new decision variables $\eta_{\omega} \in \mathbb{R}$, $\omega \in \boldsymbol{\Omega}$, equal to the value of the recourse function at optimality. Formulation F^{DW}_{Gen} now takes the form of the following mathematical program, where all second-stage constraints and costs are implicitly embedded in piecewise linear convex functions R^{ω} :

$$\min \left\{ \sum_{\omega \in \boldsymbol{\Omega}} \eta_{\omega} : \eta_{\omega} \geq R^{\omega}(\boldsymbol{\lambda},\boldsymbol{\xi}) \ \forall \omega \in \boldsymbol{\Omega}, \boldsymbol{\eta} \in \mathbb{R}^{|\boldsymbol{\Omega}|}, (32), (34), (36), (37) \right\} \tag{38}$$

A feasibility question arises that needs to be answered. A given first-stage solution $(\bar{\boldsymbol{\lambda}},\bar{\boldsymbol{\xi}},\bar{\eta}^{\omega})$ that satisfies (32), (34), (36), (37) might not be feasible for (38) if it induces an unavoidable over-production for at least one period in one scenario, or if the estimated second-stage cost $\bar{\eta}^{\omega}$ is lower than the actual cost $R^{\omega}(\boldsymbol{\lambda},\boldsymbol{\xi})$. Formally solution $(\bar{\boldsymbol{\lambda}},\bar{\boldsymbol{\xi}},\bar{\eta}^{\omega})$ is feasible for scenario ω if and only if the optimum of the following linear program is zero.

$$\begin{aligned}
f^{\omega}(\bar{\boldsymbol{\lambda}},\bar{\boldsymbol{\xi}},\bar{\eta}^{\omega}) = \min \quad & \mathbf{1}^{\top}[\tau^{\omega}_{obj},\boldsymbol{\tau}^{\omega}_{0},\boldsymbol{\tau}^{\omega}_{2}] && && (39)\\
s.t. \quad & \mathbf{P}^{\omega}\boldsymbol{p}^{\omega} && = \boldsymbol{D}^{\omega} && (40)\\
& \sum_{\omega} \bar{\boldsymbol{C}}^{\omega\top}\boldsymbol{p}^{\omega} - \tau^{\omega}_{obj} && \leq \bar{\eta}^{\omega} && (41)\\
& \boldsymbol{\Theta}^{\omega}_{0}\boldsymbol{p}^{\omega} + \boldsymbol{\tau}^{\omega}_{0} && \geq \boldsymbol{b}^{\omega}_{0} - \mathbf{A}_{0}\boldsymbol{\Lambda}\bar{\boldsymbol{\lambda}} - \boldsymbol{\Xi}_{0}\bar{\boldsymbol{\xi}} && (42)\\
& \boldsymbol{\Theta}^{\omega}_{2}\boldsymbol{p}^{\omega} + \boldsymbol{\tau}^{\omega}_{2} && \geq \boldsymbol{b}^{\omega}_{2} - \boldsymbol{\Xi}_{2}\bar{\boldsymbol{\xi}} && (43)\\
& \boldsymbol{p}^{\omega} \geq \mathbf{0} && && (44)
\end{aligned}$$

In this program, artificial variables τ^{ω}_{obj}, $\boldsymbol{\tau}^{\omega}_{0}$ and $\boldsymbol{\tau}^{\omega}_{2}$ are introduced to account for the violations of the second-stage constraints. Then the latter program mimics phase one of the simplex method where artificial variables required to be zero are allowed to be non negative. Taken apart, constraints (40) can always be

satisfied, possibly resorting to exchanges on the spot market. That is why no artificial variables are needed for them.

Since LP (39)-(44) is feasible and bounded, one can use the strong duality theorem in linear programming to express $f^\omega(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\xi}}, \bar{\eta}^\omega)$ as the optimal value of its dual program. Associating vectors of dual variables $\boldsymbol{\nu}^\omega$, μ^ω, $\boldsymbol{\rho}_0^\omega$ and $\boldsymbol{\rho}_2^\omega$ to constraints (40), (41), (42) and (43), respectively, the dual LP is as follows.

$$f^\omega(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\xi}}, \bar{\eta}^\omega) = \max \left\{ G^\omega_{\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\xi}}, \bar{\eta}^\omega}(\boldsymbol{\nu}^\omega, \mu^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) : (\boldsymbol{\nu}^\omega, \mu^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) \in \mathcal{D}^\omega \right\} \quad (45)$$

where $G^\omega_{\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\xi}}, \bar{\eta}^\omega}$ is the objective function

$$G^\omega_{\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\xi}}, \bar{\eta}^\omega}(\boldsymbol{\nu}^\omega, \mu^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) = \boldsymbol{D}^{\omega\top}\boldsymbol{\nu}^\omega + \bar{\eta}^\omega\mu^\omega + \left[\boldsymbol{b}_0^\omega - \mathbf{A}_0\boldsymbol{\Lambda}\bar{\boldsymbol{\lambda}} - \boldsymbol{\Xi}_0\bar{\boldsymbol{\xi}}\right]^\top \boldsymbol{\rho}_0^\omega + \left[b_2^\omega - \boldsymbol{\Xi}_2\bar{\boldsymbol{\xi}}\right]^\top \boldsymbol{\rho}_2^\omega$$

and $\mathcal{D}^\omega$ its feasible set

$$\mathcal{D}^\omega = \Big\{ (\boldsymbol{\nu}^\omega, \eta^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) : \mathbf{P}^{\omega\top}\boldsymbol{\nu}^\omega + \bar{\boldsymbol{C}}^\omega\mu^\omega + \Theta_0^{\omega\top}\boldsymbol{\rho}_0^\omega + \boldsymbol{\Theta}_2^{\omega\top}\boldsymbol{\rho}_2^\omega \le \mathbf{0}, \\ \boldsymbol{\nu}^\omega \in \mathbb{R}^{|\mathbf{T}|}, -1 \le \mu^\omega \le 0, \mathbf{0} \le \boldsymbol{\rho}_0^\omega \le \mathbf{1}, \mathbf{0} \le \boldsymbol{\rho}_2^\omega \le \mathbf{1} \Big\}$$

The dual LP being feasible and bounded, it admits an extreme optimal solution, and its feasibility set can be replaced with the finite set of its extreme points $\mathcal{D}^\omega_*$:

$$f^\omega(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\xi}}, \bar{\eta}^\omega) = \max \left\{ G^\omega_{\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\xi}}, \bar{\eta}^\omega}(\boldsymbol{\nu}^\omega, \mu^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) : (\boldsymbol{\nu}^\omega, \mu^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) \in \mathcal{D}^\omega_* \right\} \quad (46)$$

It follows that the condition $\eta^\omega \ge R^\omega(\boldsymbol{\lambda}, \boldsymbol{\xi})$ in (38) can be replaced with $G^\omega_{\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\xi}}, \bar{\eta}^\omega}(\boldsymbol{\nu}^\omega, \mu^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) \le 0 \ \forall (\boldsymbol{\nu}^\omega, \mu^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) \in \mathcal{D}^\omega_*$ to obtain the following Benders reformulation of model F^{DW}_{Gen}.

$$F^{DWB}_{Gen} : \min \quad \sum_\omega \eta^\omega \quad (47)$$

$$s.t. \quad \mathbf{H}\boldsymbol{\lambda} = \mathbf{1} \quad (34)$$

$$\mathbf{A}_1\boldsymbol{\Lambda}\boldsymbol{\lambda} + \boldsymbol{\Xi}_1\boldsymbol{\xi} \ge \boldsymbol{b}_1 \quad (36)$$

$$-\mu^\omega\eta^\omega + \boldsymbol{\rho}_0^{\omega\top}\mathbf{A}_0\boldsymbol{\Lambda}\boldsymbol{\lambda} + \left(\boldsymbol{\rho}_0^{\omega\top}\boldsymbol{\Xi}_0 + \boldsymbol{\rho}_2^{\omega\top}\boldsymbol{\Xi}_2\right)\boldsymbol{\xi} \ge \boldsymbol{\nu}^{\omega\top}\boldsymbol{D}^\omega + \boldsymbol{\rho}_0^{\omega\top}\boldsymbol{b}_0^\omega + \boldsymbol{\rho}_2^{\omega\top}\boldsymbol{b}_2^\omega \\ \forall \omega, (\boldsymbol{\nu}^\omega, \mu^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) \in \mathcal{D}^\omega_* \quad (48)$$

$$\boldsymbol{\xi} \ge 0, \boldsymbol{\lambda} \in \{0,1\}^{\sum_i |\mathcal{G}_i|}, \boldsymbol{\eta} \in \mathbb{R}^{|\boldsymbol{\Omega}|}$$

Note that the number of Benders cuts (48) is exponential in the size of the input data. Moreover, a single Benders cut may involve exponentially-many terms in $\boldsymbol{\lambda}$ in formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$. The use of additional variables in formulations $F_{(\overline{\boldsymbol{q}}, \boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ and $F_{(\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ allows for splitting the first and second stages, thus avoiding all terms involving $\boldsymbol{\rho}_0^\omega$. This reduces drastically the number of non-zero coefficients in formulation F^{DWB}_{Gen}.

The combined Dantzig-Wolfe and Benders reformulation of $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ (resp. $F_{(\overline{\boldsymbol{q}}, \boldsymbol{m})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ or $F_{(\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}})}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$) is denoted by $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ (resp. $F^{DWB}_{(\overline{\boldsymbol{q}}, \boldsymbol{m})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ or $F^{DWB}_{(\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$)

4.4. Row-and-column generation algorithm

Formulation F_{Gen}^{DWB} is with an exponential number of $\boldsymbol{\lambda}$-variables and Benders cuts (48). The principle is to solve dynamically its linear relaxation by combining a column generation for $\boldsymbol{\lambda}$-variables and a cutting-plane technique for Benders cuts. In this section, we describe the row-and column generation algorithm devised to solve the following *master program*, which is the linear relaxation of F_{Gen}^{DWB}.

$$[MP] : \min \left\{ \sum_{\omega} \eta^{\omega} : (34), (36), (48), \boldsymbol{\xi} \geq \mathbf{0}, \boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\eta} \in \mathbb{R}^{|\boldsymbol{\Omega}|} \right\}$$

To this aim, we introduce the *relaxed master program* $MP(\mathcal{D}_\ell)$ at *row-iteration* ℓ, obtained from MP by including the collection of Benders cuts $\mathcal{D}_\ell = (\mathcal{D}_\ell^1, \ldots, \mathcal{D}_\ell^{|\boldsymbol{\Omega}|})$ already generated in the course of the algorithm. We also define the *partial master program* $MP(\mathcal{D}_\ell, \mathcal{E}_t)$ at row-iteration ℓ and *column-iteration* t. Such program is obtained from $MP(\mathcal{D}_\ell)$ by restricting the vector of $\boldsymbol{\lambda}$-variables to the vector $\boldsymbol{\lambda}^{(t)}$ of its components whose indices are in set $\mathcal{E}_t$. Submatrices $\mathbf{H}^{(t)}$ and $\boldsymbol{\Lambda}^{(t)}$ are obtained from $\mathbf{H}$ and $\boldsymbol{\Lambda}$ by selecting the corresponding columns. Note that $MP(\mathcal{D}_\ell, \mathcal{E}_t)$ is neither a relaxation nor a restriction of MP in general, even though $MP(\mathcal{D}_\ell)$ is a relaxation of MP and thus of NOPP.

$$MP(\mathcal{D}_\ell, \mathcal{E}_t) : \min \quad \sum_{\omega} \eta^{\omega}$$

$$s.t.\ \mathbf{H}^{(t)} \boldsymbol{\lambda}^{(t)} = \mathbf{1} \tag{49}$$

$$\mathbf{A}_1 \boldsymbol{\Lambda}^{(t)} \boldsymbol{\lambda}^{(t)} + \boldsymbol{\Xi}_1 \boldsymbol{\xi} \geq \boldsymbol{b}_1 \tag{50}$$

$$-\mu^{\omega} \eta^{\omega} + \boldsymbol{\rho}_0^{\omega\top} \mathbf{A}_0 \boldsymbol{\Lambda}^{(t)} \boldsymbol{\lambda}^{(t)} + \left(\boldsymbol{\rho}_0^{\omega\top} \boldsymbol{\Xi}_0 + \boldsymbol{\rho}_2^{\omega\top} \boldsymbol{\Xi}_2 \right) \boldsymbol{\xi} \geq \boldsymbol{\nu}^{\omega\top} \boldsymbol{D}^{\omega} + \boldsymbol{\rho}_0^{\omega\top} \boldsymbol{b}_0^{\omega} + \boldsymbol{\rho}_2^{\omega\top} \boldsymbol{b}_2^{\omega}$$
$$\forall \omega, (\boldsymbol{\nu}^{\omega}, \mu^{\omega}, \boldsymbol{\rho}_0^{\omega}, \boldsymbol{\rho}_2^{\omega}) \in \mathcal{D}_\ell^{\omega} \tag{51}$$

$$\boldsymbol{\xi} \geq 0, \boldsymbol{\lambda}^{(t)} \in \mathbb{R}_+^{|\mathcal{E}_t|}, \boldsymbol{\eta} \in \mathbb{R}^{|\boldsymbol{\Omega}|} \tag{52}$$

The cutting-plane algorithm given in Algorithm 1 is the outer loop of the column-and-row generation procedure to solve MP. It calls iteratively the column generation algorithm given in Algorithm 2 to solve relaxed master programs $MP(\mathcal{D}_\ell)$. At each iteration, the objective function of such programs is improved until no more improvement is possible, thus leading to a tight relaxation. At the initial iteration, it starts with a minimal set of columns and Benders cuts (see line 1 of Algorithm 1). Many strategies can be designed in this purpose. To keep the partial master programs small, our implementation choice is to start without any Benders cuts, and with a single path for each nuclear unit, namely the shortest path in each transition graph with original costs. The main loop starts by solving the current relaxed master program $MP(\mathcal{D}_\ell)$ (see line 3), thus obtaining an optimal solution $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$. If it is feasible for MP, then it is also an optimal solution for it as $MP(\mathcal{D}_\ell)$ is a relaxation of MP with the same objective function. The Benders *separation problem* checks whether all

Algorithm 1: Outer loop of the column-and-row generation algorithm to solve MP. This cutting-plane component iteratively calls the column generation algorithm to solve relaxed master programs $MP(\mathcal{D}_\ell)$, which are improved at each iteration until the relaxation is tight.

1 $\ell \leftarrow 0$; $t \leftarrow 0$; Initialize $\mathcal{E}_0$ and $\mathcal{D}_0$; $n_c \leftarrow |\mathcal{E}_0|$
2 **repeat**
3 Solve $MP(\mathcal{D}_\ell)$
4 Let $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$ be the optimal solution
5 $NewCuts \leftarrow false$; $\mathcal{D}_{\ell+1} \leftarrow \mathcal{D}_\ell$
6 **foreach** $\omega \in \boldsymbol{\Omega}$ **do**
7 Solve (45) to compute $F^\omega(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$
8 **if** $F^\omega(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*) > 0$ **then**
9 Let $(\boldsymbol{\nu}^{\omega*}, \mu^\omega, \boldsymbol{\rho}_0^{\omega*}, \boldsymbol{\rho}_2^{\omega*})$ be the optimal solution of (45)
10 $\mathcal{D}_{\ell+1} \leftarrow \mathcal{D}_{\ell+1} \cup \{(\boldsymbol{\nu}^{\omega*}, \mu^\omega, \boldsymbol{\rho}_0^{\omega*}, \boldsymbol{\rho}_2^{\omega*})\}$
11 $NewCut \leftarrow true$
12 $\ell \leftarrow \ell + 1$
13 **until** $NewCuts = false$
14 **return** $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$

constraints (48) are satisfied or not. It decomposes for each scenario $\omega \in \boldsymbol{\Omega}$ into one independent subproblem, which is to compute $F^\omega(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \eta^{\omega*})$ using LP (45) (see line 7). If $F^\omega(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \eta^{\omega*}) > 0$, then the corresponding Benders cut is violated by solution $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$. For each scenario satisfying this condition, the associated cut is added to the formulation, thus defining $\mathcal{D}_{\ell+1}$ (see line 9). The algorithm iterates solving $[MP(\mathcal{D}_{\ell+1})]$ until no more violated cuts can be found. In the latter case, $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$ is a feasible and optimal solution of MP, and the algorithm stops.

In order to solve each relaxed master program $MP(\mathcal{D}_\ell)$ involved in the inner loop, Algorithm 2 solves to optimality the partial master program $MP(\mathcal{D}_\ell, \mathcal{E}_t)$ (with the restricted set of $\boldsymbol{\lambda}-$variables $\mathcal{E}_t$, see line 2). Let us consider an optimal solution $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$, and corresponding dual values $(\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)$ associated with constraints (49), (50) and (51), respectively. Linear programming theory tells us that $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$ is an optimal solution of $MP(\mathcal{D}_\ell)$ if the reduced cost (*w.r.t.* $(\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)$) of all $\boldsymbol{\lambda}$-variables in $MP(\mathcal{D}_\ell)$ are non-negative. For the sake of readability, we use the following generic form of the reduced cost for variable λ_{ig}, $i \in \mathbf{I}$, $g \in \mathcal{G}_i$ [3]: $\sum_{e \in \boldsymbol{E}_i} \tilde{c}_e(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)\Lambda_e^g - \pi_i^*$, where coefficient $\tilde{c}_e(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)$ is the contribution of $\boldsymbol{\Lambda}^g$ in the dual LP of $MP(\mathcal{D}_\ell)$. A remarkable feature of this expression is that it is only related to nuclear unit i. It follows that the *pricing*

[3] The detailed expression of the reduced cost of variable λ_{ig}, $i \in \mathbf{I}$, $g \in \boldsymbol{\mathcal{G}}_i$ is $-\boldsymbol{\pi}^{*\top}\mathbf{H}^g - \left(\boldsymbol{\mu}^{*\top}\mathbf{A}_1 + \sum_{\omega \in \boldsymbol{\Omega}} \sum_{(\boldsymbol{\nu}^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega) \in \mathcal{D}_*^\omega} \boldsymbol{\sigma}_{\boldsymbol{\nu}^\omega, \boldsymbol{\rho}_0^\omega, \boldsymbol{\rho}_2^\omega}^{*\top} \boldsymbol{\rho}_0^{\omega\top}\mathbf{A}_0\right)\boldsymbol{\Lambda}^g$

Algorithm 2: Inner loop of the column-and-row generation algorithm to solve MP. This column generation component solves partial master programs $[MP(\mathcal{D}_\ell, \mathcal{E}_t)]$, columns being iteratively added until $MP(\mathcal{D}_\ell)$ is solved to optimality.

1 **repeat**
2 Solve $MP(\mathcal{D}_\ell, \mathcal{E}_t)$
3 Let $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$ and $(\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \sigma^*)$ be the primal and dual optimal solutions, respectively
4 $NewCols \leftarrow false$; $\mathcal{E}_{t+1} \leftarrow \mathcal{E}_t$
5 **foreach** $i \in \mathbf{I}$ **do**
6 Solve $[Pricing_i(\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)]$
7 Let $\boldsymbol{\delta}^*$ be the optimal solution
8 **if** $\sum_{e \in E_i} \tilde{c}_e(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)\delta_e - \pi_i < 0$ **then**
9 $n_c \leftarrow n_c + 1$; $NewCol \leftarrow true$
10 $\mathcal{E}_{t+1} \leftarrow \mathcal{E}_{t+1} \cup \{n_c\}$
11 $(\boldsymbol{\Lambda}^{(t+1)})^{n_c} \leftarrow \boldsymbol{\delta}^*$; $(\mathbf{H}^{(t+1)})^{n_c} \leftarrow \boldsymbol{\epsilon}_i$
12 **if** $NewCols = true$ **then**
13 $t \leftarrow t + 1$
14 **until** $NewCols = false$
15 **return** $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$

problem, which is to find the minimum reduced cost $\boldsymbol{\lambda}$-variable, decomposes for each unit $i \in \mathbf{I}$ into one subproblem $[Pricing_i(\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)]$ (solved in line 6) is as follows.

$$
\begin{aligned}
[Pricing_i(\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)] : \min & \sum_{e \in \boldsymbol{E}_i} \tilde{c}_e(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)\delta_e - \pi_i^* \\
s.t \quad & \delta_{(u,v)} - \sum_{(v,u) \in E_i} \delta_{(v,u)} = 0 \qquad \forall u \in V_i \backslash \{s(\boldsymbol{G}_i), t(\boldsymbol{G}_i)\} \\
& \sum_{(s(\boldsymbol{G}_i),v) \in \boldsymbol{E}_i} \delta_{(s(\boldsymbol{G}_i),v)} = 1 \\
& \delta_e \in \{0,1\} \qquad \forall e \in \boldsymbol{E}_i
\end{aligned}
$$

Subproblem $[Pricing_i(\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)]$ seeks for a shortest path in the transition graph of unit i with modified costs on the arcs. If negative reduced cost $\boldsymbol{\lambda}$-variables are found, they are added to the formulation, thus defining $\mathcal{E}_{t+1}$. More precisely, line 11 appends the corresponding vector to matrix $\boldsymbol{\Lambda}^{(t+1)}$ and registers this column into the set of variables for unit i (by appending the i^{th} canonical vector $\boldsymbol{\epsilon}_i$ to matrix $\mathbf{H}^{(t+1)}$). The algorithm iterates solving $MP(\mathcal{D}_\ell, \mathcal{E}_{t+1})$ until no negative reduced cost $\boldsymbol{\lambda}$-variable is found. In the latter case, $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$ is an optimal solution of $MP(\mathcal{D}_\ell)$.

4.5. Obtaining integer feasible solutions

To obtain feasible solutions for NOPP, formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ can be solved directly with an MIP solver.

The master program MP, solved using the row-and-column generation algorithm described in Section 4.4, is a building block for several methods. A first option is to use an exact method based on a branch-and-price-and-cut procedure (see *e.g.*, [5]) to solve the NOPP from the master program MP. Schematically it is a branch-and-bound algorithm in which the dual bound used at each node of the search tree is MP augmented with branching constraints. Its computation is quite similar to Algorithm 1, where the relaxed master program also contains branching constraints. When solution $(\boldsymbol{\lambda}^*, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$ of the relaxation does not satisfy integrality requirements (*i.e.*, some original variable $\delta_e = \sum_{g \in \mathcal{G}_i} \Lambda_e^g \lambda_{ig}^*$ is not integer), one creates two child nodes, in which additional branching constraints $\sum_{g \in \mathcal{G}_i} \Lambda_e^g \lambda_{ig} = 0$ and $\sum_{g \in \mathcal{G}_i} \Lambda_e^g \lambda_{ig} = 1$ are imposed, respectively. Unfortunately, this exact solving procedure is not appropriate to deal with the NOPP as the computation time required is prohibitive for large-scale instances.

A second option is to use branch-and-price-based heuristics (see [18] for the presentation of several heuristics). The *pure diving heuristic* is a greedy algorithm: it first solves MP. If the obtained solution is not integer, a greedy solution is constructed by choosing a branch in the branch-and-price search tree following a given criterion. At the given branch, the master program is solved using the column generation algorithm. The heuristic stops when an integer solution is found, or when the current node of the search tree is infeasible. Following the implementation described in [18], we choose the $\boldsymbol{\lambda}-$variable whose value is closest to 1, and branch by fixing its value to 1. In the case of a column-and-row generation algorithm the principle is similar except MP is solved by column-and-row generation before each greedy selection of a branch in the branch-and-price-and-cut tree.

This heuristic often suffers from its myopic behavior and can be improved using *least discrepancy search* instead of a purely greedy search. The idea is to explore a larger part of the search tree by allowing limited backtracking. Given a maximum discrepancy parameter $maxDiscrepancy$, the algorithm explores paths of the search tree that are obtained by applying the greedy criterion of the pure diving procedure except for, at most, $maxDiscrepancy$ branching choices. The search is also limited by forcing the use of the greedy criterion at nodes whose depth in the search tree is larger than parameter $maxDepth$.

We also investigated the use of the *restricted master heuristic*, also called *price-and-branch* heuristic. In the context of dynamic generation of columns only, the principle is to first solve the linear relaxation of the reformulated problem using column generation, thus obtaining a *restricted master program* with a subset of all the columns. Integrality requirements are then reintroduced into the current formulation, and the restricted master program is solved as a static MIP, *i.e.*, without generating new columns. In the context of column-and-row generation, this algorithm must be customized to account for the dynamically generated constraints. Our strategy is to combine the price-and-branch heuristic

with the *branch-and-Benders-cut* algorithm [7], leading to the so-called *price-and-branch-and-Benders-cut* algorithm. The procedure first solves formulation MP using Algorithm 1, thus generating columns and Benders cuts necessary to solve the linear relaxation of the considered reformulations. Then, it yields a partial master program $MP(\mathcal{D}_\ell, \mathcal{E}_t)$. The algorithm is a heuristic in the sense that no new columns are generated after the root node processing. However, in order to obtain a feasible solution, one needs to ensure that all Benders cuts are satisfied. Formally, we solve problem $MP(\mathcal{D}_*, \mathcal{E}_t)$, where $\mathcal{D}_*$ is the set of extreme points of the dual LP (39)-(44), with additional integrality restrictions $\boldsymbol{\lambda}^{(t)} \in \{0,1\}^{|\mathcal{E}_t|}$. The corresponding formulation is solved using an MIP solver, starting with the restricted set of cuts $\mathcal{D}_\ell$ explicitly included. Whenever an integer candidate solution $(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \boldsymbol{\eta}^*)$ is found during the search, the separation problems (45) are solved via the solver's callback interface, for all $\omega \in \boldsymbol{\Omega}$. If violated Benders cuts are identified, *i.e.*, $F^\omega(\boldsymbol{\lambda}^{(t)*}, \boldsymbol{\xi}^*, \eta^{\omega *}) > 0$ for some ω, they are dynamically added to the formulation and the candidate solution is rejected.

The quality of the solutions obtained using the price-and-branch heuristic improves as the number and the diversity of columns in the restricted master program increases. The *diving with sub-MIPing heuristic* exploits this idea by first calling the pure diving heuristic, and second solving a restricting master composed of all the columns generated during the diving (with integrality restrictions). This method can suffer from a large number of columns, leading to elevated computation times.

5. Computational results

5.1. Instances

The proposed formulations and solution approaches are evaluated on a real data-set of the french electricity production with a time horizon ranging from January 2015 to December 2018. The data-set is composed of 58 nuclear units and 84 other sources accounting for non-nuclear units, and exchanges on the market spot.

The stochastic data are given through 5 sets of 96 scenarios of demand and non-nuclear unit characteristics (production bounds and leasing costs). From this original data-set, we derive several NOPP instances by reducing the number scenarios and possibly scaling down the fleet. From each set of scenarios and given an integer S, a dedicated Scenario Clustering Library provided by EDF generates S aggregated scenarios. This library clusters similar original scenarios using a norm based on a heuristic cost evaluation of demand satisfaction accounting for a set of random parameters attached to each scenario (demand, availability of non-nuclear units, prices/capacities on the electricity spot market).

The fleet is scaled down when needed by considering a subset of the nuclear units as non-nuclear units, thus reducing the nuclear share in the instances while the other data are not changed. Such instances are referred to as *scaled-down instances* in the sequel. To maintain local power plant scheduling constraints valid, we first select a subset of nuclear power plants that is kept unchanged in the scaled-down instances. In the data-set, a baseline planning defines, for each of the remaining nuclear units i, outage periods during which its production

is set to zero, while they can produce between zero and $\overline{P}_{it}$ for other periods $t \in \mathbf{T}$. The corresponding constraints on modulation and fuel level are relaxed, and removed from all scheduling constraints. Last the leasing cost of the new non-nuclear units is derived from real data relative to EDF nuclear units.

Instances are classified into instance types, labeled using the following pattern *pAiB-C.D.EsS* where:

- A is the number of nuclear units. $A = 58$ corresponds to EDF current nuclear fleet, $A = 22$ and 40 are considered for scaled-down instances.
- B is the instance type tag number. Each unique value of B corresponds to a selection of a subset of nuclear units in the real data-set that are kept unchanged, and a number of aggregated scenarios.
- $C \in \{0,1\}$ is the number of 6-unit power plants in type B.
- $D \in \{0, 1, \ldots, 8\}$ is the number of 4-unit power plants in type B.
- $E \in \{0, 1, \ldots, 10\}$ is the number of 2-unit power plants in type B.
- S is the number of aggregated scenarios in type B.

Note that instances inside a same type differ by the set of considered scenarios.

The formulation of the first stage problem described in Section 3.2 has strong linear relaxation, but it comes at the price of a very large number of constraints and variables. For real instances p58i0-1.8.10sS (for any number of scenario S), the transition graphs lead to around 120.000 binary arc variables, 30.000 flow conservation constraints (corresponding to the number of nodes in all graphs) and 3.000 scheduling constraints binding outage arcs from different graphs. Hence, anticipating that it will not be possible to solve the formulation directly with a MIP solver for a large number of scenarios, we will study the use of Dantzig-Wolfe decomposition-based heuristics combined or not with Benders' decomposition. The Dantzig-Wolfe reformulation is not used here to obtain an improvement in the relaxation but for its ability to decompose the problem into smaller subproblems solved iteratively.

Tests are carried out on a Linux machine equipped with 2×12-core Haswell Intel Xeon E5-2680 v3 CPUs with 128 Go of RAM. Modeling and solving are done using BaPCod which is a black-box framework dedicated to solve MIPs using reformulation techniques such as Dantzig-Wolfe and Benders decompositions [20]. At most one column for each nuclear unit is added to the partial master program at each iteration. To improve the convergence of the column generation procedure, we use stabilization by automatic smoothing the dual variables of the master program, as described in [15]. We use `Lemon` library for modeling the graph structure and shortest path algorithm, and `Boost 1.56` library for parallelization of Benders subproblems (the solution of column generation subproblems is sequential). The MIP solver used is Cplex solver 12.7.1. In order to have a fair comparison between the different solution approaches we limit to 6 the number of threads used by the MIP solver in the tests of Sections 5.3 and 5.5. In Section 5.6 the solver is used with default settings.

5.2. Comparing arc flow-based formulations

In this section a few preliminary results is provided to show the comparative performance of the first two proposed MIP formulations, namely $F_{(\underline{q},\overline{q})}(\boldsymbol{\delta}, \boldsymbol{p})$

given in Section 3.4 and $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ in Section 3.5. Table 2 shows the computation time in seconds required to solve the LP relaxation of each formulation for the simplified NOPP instances with 22 nuclear units. The difference in terms of performance is quite significant in favor of formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$. Then the latter formulation should lead to better performance when used by a linear solver at each node of a branch and bound tree or at each column generation iteration.

Instance type	$F^{LP}_{(\underline{q},\overline{q})}(\delta, p)$	$F^{LP}(\delta, s, p)$
p22i0-1.4.0s1	142,2	55,8
p22i1-0.1.9s1	129,2	26,0
p22i2-1.3.2s1	182,4	47,6
p22i3-0.4.3s1	159,2	35,8
p22i4-0.4.3s1	102,0	28,0

Table 2: Comparative computation time (in seconds) to solve the LP relaxation of $F_{(\underline{\boldsymbol{q}},\overline{\boldsymbol{q}})}(\boldsymbol{\delta}, \boldsymbol{p})$ and $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$.

5.3. Comparing heuristics to solve deterministic instances

In this section the aim is to compare different solution approaches on simplified deterministic instances to select the most promising approach anticipating real size stochastic instances. Note that the cut generation procedure is not needed to solve a deterministic instance, nor is the use of capacity variables defined in (16)-(17). It would only add variables in the master of the column generation procedure without any benefit. Hence, we compare the performance using the original integer formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$, as defined in Section 3.5, solved directly with Cplex solver or using Dantzig-Wolfe reformulation $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$ solved by a column generation algorithm, as described in Section 4.4, followed by a heuristic to obtain a feasible solution for NOPP.

Note that the work presented in this article focuses on solving efficiently the linear relaxation of several formulations to find a good primal solution in terms of quality through a heuristic. Hence we used the generic column generation-based heuristics of the literature without any further analysis of the internal solving process for improvement. Such work is beyond the scope of this article. The following benchmark of heuristics, described in Section 4.5, is considered:

- Price-and-branch: First price, i.e., solve formulation $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$ by column generation at the root node, and then branch, i.e., enforce integrity constraints and solve the resulting restricted master program to optimality using Cplex solver.
- Pure diving: Formulation $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$ is solved by column generation followed by a diving heuristic with maxDiscrepancy=0 and maxDepth=0, i.e., greedy construction of a solution alternating branching in the branch-and-price tree and solving the master problem.
- Diving23: Formulation $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$ is solved by column generation followed by a diving heuristic with maxDiscrepancy=2 and maxDepth=3.

- Pure diving + price-and-branch: Formulation $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$ is solved by column generation followed by a combination of a pure diving heuristic before price-and-branch. Then the initial solution from pure diving could lead to generate improving columns, thus possibly reducing the number of visited nodes during the price-and-branch.

Comparative results for simplified deterministic instances using the column generation based heuristics benchmark are presented in Table 3, which features for each instance type:

- $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ - Int.gap: average integrity gap $\frac{opt-b}{opt}$ between the integer optimum *opt* and the optimal linear bound b.
- $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ - CPU(s): average computation time in seconds to solve the mixed integer original formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ to optimality with Cplex solver;
- for each heuristic in the benchmark:
  - gap: average gap $\frac{p-opt}{p}$ between the integer value p found by the heuristic and the integer optimum *opt* found by $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$;
  - CPU(s): average computation time in seconds.

Note first that for this set of instances the optimal value is obtained with $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ solved directly by Cplex solver. Therefore, the gap being zero is not shown and replaced by the integrity gap in Table 3. Note also that b is the optimal value of both the linear relaxation $F^{LP}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ and the Dantzig-Wolfe reformulation $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$.

	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$		Price-and-branch		Pure diving		Diving23		Pure diving +Price-and-branch	
Instance type	Int.gap(%)	CPU(s)	gap(%)	CPU(s)	gap(%)	CPU(s)	gap(%)	CPU(s)	gap(%)	CPU(s)
p22i0-1.4.0s1	0,67	68,2	0,06	54,4	0,06	97,4	0,06	680,2	0,05	64,4
p22i1-0.1.9s1	1,03	67,8	0,20	40,4	0,30	64,2	0,30	429,2	0,13	66,4
p22i2-1.3.2s1	0,59	71,4	0,01	34	0,10	44	0,06	336,2	0,00	40,8
p22i3-0.4.3s1	1,26	67,4	0,61	30,2	0,50	85	0,48	445,4	0,05	94,2
p22i4-0.4.3s1	0,68	53,5	0,01	18,75	0,05	43,5	0,04	210,75	0,01	42,75
p40i0-1.8.1s1	1,23	282,6	0,91	186,8	$13{,}97^a$	391,8	$13{,}97^a$	3494,6	0,16	489,6
p40i1-0.5.10s1	0,86	151,8	0,95	110	$5{,}40^b$	185,2	$5{,}20^b$	1848,2	0,21	216,6
p40i2-1.5.7s1	1,26	1052	0,46	406,4	0,83	403,8	0,70	4587,6	0,16	919,4

Table 3: Comparing performance of exact solutions and solutions obtained using the column generation based heuristics benchmark for simplified deterministic instances.

The performance results presented in Table 3 correspond to average with respect to sets of five instances. The average reflects quite well the performance results of each instance in the corresponding set for all sets of instances but for two sets relative to 40 nuclear unit instances. For these two latter instances, gaps in Table 3 appear with superscript ^{a} (resp. ^{b}) to indicate cases where a large variability in the gaps have been observed, namely 3 (resp. 1) out of 5 solutions are with a gap around 20-25%, whereas 2 (resp. 4) out of 5 solutions are with a gap close from 1%. In other words, diving heuristics either find solutions with a gap less than 1% or around 20-25%. An explanation for such behavior is twofold. First binding constraints are active on 40 unit instances while they are inactive on 22 nuclear unit instances. Second columns do not have coefficients in the

objective function, thus leading to poor branching choices. Anticipating solving large size instances, the computation time should be less than that obtained for solving $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ directly by Cplex solver.

The ranking of the different heuristics relative to the computation time required to solve $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ directly by Cplex solver is as follows: diving23 is extremely slow, pure diving heuristic is slow, pure diving + price-and-branch is close to Cplex solver, and price-and-branch outperforms Cplex solver. The price-and-branch heuristic always finds a solution with a very good gap less than 1% in average and with a computation time 30-50% less than that obtained by Cplex solver. Note that in terms of quality, pure diving + price-and-branch finds excellent solutions but with a computation time close to that obtained with Cplex solver.

This benchmark suggests to try solving real instances directly with Cplex solver whenever possible and to use price-and-branch heuristic otherwise.

5.4. Model validation

The main thrust of this work is that outage dates computed using the NOPP formulation will lead to operational savings when accounting for an increased number of scenarios.

As mentioned in Section 1, the operational outage planning process is a multi-stage procedure involving successive re-optimizations of outage dates and power productions on a rolling horizon. Designing a code to emulate this multi-stage process to evaluate a first-stage solution over a set of validation scenarios is beyond the scope of this paper. We use a dedicated tool developed at EDF to evaluate solutions in the limited framework of a two-stage process (consistent with the structure of NOPP). This library, referred to as *Checker*, takes as input a scenario and a given nuclear outage plan, and optimizes the production of the units while meeting the demand over the time horizon, thus emulating a NOPP second stage. This is modeled as a simple linear program, which allows refining the time discretization to six periods per day (instead of one per week used in the optimization models).

The cost of each first-stage solution is evaluated as the expected second-stage cost calculated with the checker, over 96 scenarios. For each of the 25 (resp. 15) scaled-down instances with 22 (resp. 40) nuclear units, we computed first-stage solutions with $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ and $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$. Recall that $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ is directly solved with Cplex solver as defined in Section 3.5, while $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$ is to solve the linear relaxation with column generation and to use the Price-and-branch heuristic selected in section 5.3. The number of aggregated scenarios considered for the optimization ranges in $S \in \{1, 5, 10, 15\}$. This allows us to estimate the savings obtained when using a given method with a given number of aggregated scenarios, compared to the deterministic case, i.e., with one aggregated scenario. More precisely, the savings correspond to the expected cost of the first-stage solution obtained with a method and S scenarios minus the expected cost of the solution obtained with $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ and $S = 1$. In Table 4 statistics relative to the savings are reported over the whole set of simplified 22-unit instances (resp. 40-unit instances), denoted by p22* (resp. p40*). Rows "avg. savings"

and "stddev. savings" respectively show the average and standard deviation, over the considered instance type, of savings evaluated by the Checker over the 96 original scenarios. Note that the objective functions of the scaled-down instances is considerably increased compared to the original ones: this is a side effect of virtually converting nuclear units to thermal ones. To better emphasize the benefit of using more scenarios, absolute savings are reported. The order of magnitude of an absolute difference of 1 monetary unit is, here, 0.001% in relative difference. A one-hour time limit is imposed for each run. Within this time limit, Cplex solver ($F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$) converges to optimality for all runs up to $S = 10$. For 22 units and $S = 15$, optimality was not proven but the optimality gap is less than 0.15%. We do not report results for instances with 40 units and $S = 15$, because both methods failed at finding feasible solutions for most instances.

Instance type	Statistic	Method	S=1	S=5	S=10	S=15
p22*	avg. savings	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	0.00	7.28	8.62	11.80
		$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	-3.64	4.75	2.22	4.62
	stddev. savings	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	0.00	6.65	5.79	6.47
		$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	3.72	7.43	7.85	6.10
p40*	avg. savings	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	0.00	21.47	23.49	-
		$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	-9.83	0.53	8.42	-
	stddev. savings	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	0.00	15.67	15.94	-
		$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	9.85	16.97	17.02	-

Table 4: EDF Checker evaluation of optimal and heuristic solutions, absolute difference w.r.t. the planning computed on deterministic instances.

We observe that the expected savings of optimal solutions ($F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$) increase with the number of aggregated scenarios. The same trend is obtained with heuristic solutions even if the corresponding expected savings are not as good as the one obtained with optimal solutions, they globally increase with the number of aggregated scenarios. However, the case of $S = 5$ with 22 units appears as an outlier that can be explained by the variability in terms of quality of the heuristic solutions.

5.5. Comparing formulations for simplified stochastic instances

This section aims at selecting, on simplified stochastic instances, the most promising approaches to solve real size stochastic instances. Several formulations and solution approaches might be efficient on stochastic instances depending on the number of aggregated scenarios taken into account. Note first that linear relaxation is a major component to any solution approach. Then to limit the number of experiments to be included in the article, the comparative results for stochastic scaled-down instances are performed with the linear relaxation of the problem using the following benchmark of formulations along with their solution approaches.

- $F^{LP}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$: Linear relaxation of the MIP formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ described

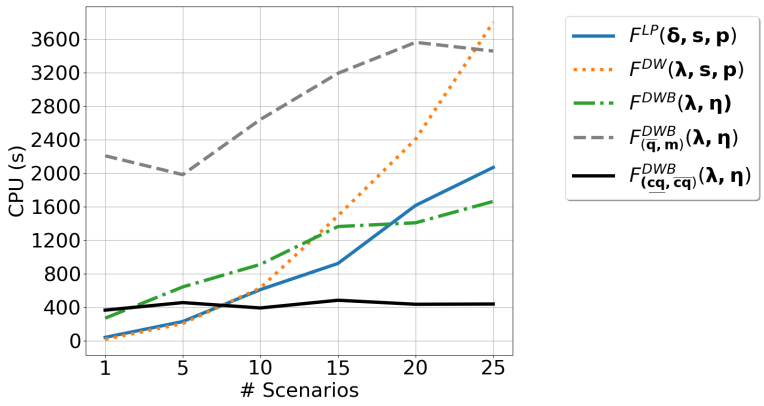

Figure 4: Comparison of computation times for solving the LP relaxation of 22 nuclear unit stochastic instances using a benchmark of formulations.

in Section 3.5 and solved directly using Cplex solver.

- $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$: the Dantzig-Wolfe reformulation, as described in Section 4.2, solved by column generation as described in Section 4.4.
- $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$: the Dantzig-Wolfe and Benders reformulation as described in Section 4.2 and 4.3 and solved by column-and-cut generation algorithms as described in Section 4.4.
- $F^{DWB}_{(\bar{\boldsymbol{q}}, \boldsymbol{m})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$: The Dantzig-Wolfe and Benders reformulation using intermediary variables $\bar{\boldsymbol{q}}$ and $\boldsymbol{m}$ linking first and second stage as described in Section 3.6. and solved by column-and-cut generation algorithms as described in Section 4.4.
- $F^{DWB}_{(\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$: the Dantzig-Wolfe and Benders reformulation using intermediary variables $\underline{\boldsymbol{cq}}$ and $\overline{\boldsymbol{cq}}$ linking first and second stage as described in Section 3.6 and solved by column-and-cut generation algorithms as described in 4.4.

Figure 4 shows the average computation time on sets of 25 stochastic instances, five for each possible structure with 22 nuclear units. The stochastic instances are the same as the 22 nuclear unit deterministic instances presented in Table 3, but for the number of scenarios S which ranges from 1 to 25.

Approaches combining Dantzig-Wolfe and Benders reformulations $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$, $F^{DWB}_{(\bar{\boldsymbol{q}}, \boldsymbol{m})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ and $F^{DWB}_{(\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ solved through column-and-cut generation are slower on instances with few scenarios. Whereas approaches without Benders decomposition $F^{LP}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ and $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$ are faster, but with rapid deterioration of performance as the number of scenarios increases. To be more specific, the solution time of $F^{LP}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ (resp. $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$) ranges from less than 60 seconds for one scenario to more than 2000 seconds for solving $F^{LP}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ (resp. 3600 seconds for solving $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$) with 25 scenarios.

Beyond 15 scenarios, $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ and $F^{DWB}_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ become more efficient than $F^{LP}(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$. Interestingly the solution time increases really slowly – indeed is almost constant – with respect to the number of scenarios for $F^{DWB}_{(\underline{cq},\overline{cq})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$. This clearly shows that the reformulation with cumulative capacity variables is the most efficient to solve the LP relaxation for stochastic instances with a large number of scenarios. The computation time appears to be linear in the number of scenarios up to 15 using $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ (green plot in Figure 4) while it is almost constant using $F^{DWB}_{(\underline{cq},\overline{cq})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ (black plot in Figure 4). It is interesting to check in more details which components of the column-and-cut algorithms are making the difference. We consider a component-wise comparison of computation times to evaluate more closely the performances of each solution approach. Table 5 provides for each of the three formulations solved through cut-and-column generation the following entries evaluated on an average of the 22 nuclear unit stochastic instances :

- S is the number of aggregated scenarios;
- For column generation (resp. cut generation), denoted by $ColGen$ (resp. $CutGen$), $\#it$ is the number of iterations and $\#Col$ (resp. $\#Cuts$) is the total number of columns (resp. Cuts) generated.
- TotalLP is the total time spent solving the linear relaxation
- SolMaster is the total time spent solving the restricted master problem during column generation,
- SolSep is the time spent solving the separation subproblems and generating cuts which involves the projection,
- UpPric is the time to update arc costs with dual variables coming from the master solution,
- UpSep + SolPric is the sum of the time spent updating first-stage decisions in the separation subproblems and solving the shortest path problem – i.e., pricing subproblem – and generating new columns. The two components have been added together as they represent a really small part of the time spent in other steps.

First note that the number of generated columns variations does not seem to have a clear link with the number of aggregated scenarios for any formulation and for a given number of scenarios all formulations require a similar number of cuts and cut generation iterations. Second it appears that $F^{DWB}_{(\overline{\boldsymbol{q}},\boldsymbol{m})}(\lambda, \eta)$ requires twice as many column generation iterations compared to others formulations and far more columns. This deeply affects the performance of the latter formulation. Formulation $F^{DWB}_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}$ requires the same number of column generation iterations as formulation F^{DWB} does but 1.3 times more columns, which explains the relative low efficiency of formulation $F^{DWB}_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}$ for few scenarios. Then an interesting question that arises is why the solving time is constant w.r.t. the number of scenarios with intermediary variables $(\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}})$, while it is linear without.

Let us look at the time spent in each component of the algorithm. First note that the time spent in components SolMaster and UpPric is larger in a deterministic setting with intermediary variables $(\overline{\boldsymbol{q}}, \boldsymbol{m})$ or $(\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}})$. This is

		ColGen		CutGen		Computation time component-wise (s)				
S	Formulation	# It	# Col	# It	# Cuts	TotalLP	SolMaster	SolSep	UpPric	UpSep + SolPric
1	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	410	1590	90	89	159,8	27,7	80,6	43,0	2,0
	$F^{DWB}_{(\underline{cq},\overline{cq})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	433	2297	89	88	370,4	151,5	67,2	116,1	3,4
	$F^{DWB}_{(\overline{q},m)}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	868	5010	88	87	1510,2	1182,0	92,0	196,1	6,2
5	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	355	1539	52	245	445,7	78,9	215,3	124,0	6,2
	$F^{DWB}_{(\underline{cq},\overline{cq})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	365	2228	51	240	374,5	145,1	71,1	108,0	5,8
	$F^{DWB}_{(\overline{q},m)}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	747	4613	52	242	1274,0	937,0	80,6	189,4	8,8
10	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	342	1545	43	405	715,7	102,1	365,6	193,9	9,3
	$F^{DWB}_{(\underline{cq},\overline{cq})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	346	2242	44	406	421,8	157,6	87,7	109,1	8,6
	$F^{DWB}_{(\overline{q},m)}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	685	4391	44	405	1411,0	1050,0	94,9	178,3	10,8
15	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	360	1671	42	590	1036,9	159,3	499,3	288,7	12,7
	$F^{DWB}_{(\underline{cq},\overline{cq})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	362	2283	41	576	526,0	125,0	105,0	131,7	3,9
	$F^{DWB}_{(\overline{q},m)}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	729	4453	41	579	1878,6	1460,0	105,8	193,6	14,5

Table 5: Comparative performance for solving the LP relaxation for 22 nuclear unit stochastic instances using the three Danzig-Wolfe and Benders formulations through cut-and-column generation.

consistent with the larger number of generated columns. However the time spent in components SolSep and UpPric is almost constant with intermediary variables $(\underline{\boldsymbol{cq}}, \overline{\boldsymbol{cq}})$ while it grows w.r.t. the number of scenarios without. The rationale behind is that without intermediary variables one needs to perform a projection as described in Section 3.6. The number of coefficients, involved in the generated cuts rewritten with the original variables or in the pricing update to add new cuts with dual values, increases rapidly with the number of aggregated scenarios (and hence the number of generated cuts) taken in account. This results in a significant increase in the time spent in components SolSep and UpPric, i.e., components of the algorithms where projection is performed, which is 82% of the total time increase. This clearly answers the question.

5.6. Solving the real large-scale instances

The aim in this section is to perform a final comparative evaluation among formulations and solution approaches, which have passed previous evaluations with reasonable chances to solve large size instances. Contrary to Section 5.5, we are looking to mixed integer solutions. The considered benchmark is:

- $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$: Original MIP formulation solved directly using Cplex solver.
- $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$: Dantzig-Wolfe reformulation solved by column generation, followed by the best promising heuristic from Section 5.3, namely price-and-branch as described in Section 4.5.
- $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$: Dantzig-Wolfe and Benders reformulation solved by column-and-cut generation algorithms, followed by the price-and-branch as described in Section 4.5.
- $F^{DWB}_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$: the Dantzig-Wolfe and Benders reformulation using intermediary variables $\underline{\boldsymbol{cq}}$ and $\overline{\boldsymbol{cq}}$ and solved by column-and-cut generation algorithms, followed by the price-and-branch as described in Section 4.5.

		LP relaxation					LP + heuristic (s)				
S	Formulation	# ColGen It	# Col	# CutGen It	# Cuts	TotalLP	SolMaster	SolSep	UpPric	UpSep + SolPric	Gap
1	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	0.49
	$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	79	1768	-	-	255	180	-	52	2	0.57
	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	1146	7247	723	722	6462	3993	22332	1523	94	0.57
	$F^{DWB}_{\underline{cq},\overline{cq}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	1021	10035	768	767	9740	8079	10819	772	113	0.66
5	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	0.71
	$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	79	1925	-	-	5199	4702	-	394	4	0.87
	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	738	6168	227	1111	8032	3511	19297	2485	177	1.04
	$F^{DWB}_{\underline{cq},\overline{cq}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	691	9454	227	1115	8570	7250	10309	279	283	0.94
10	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	0.89
	$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	69	1845	-	-	24268	23112	-	927	8	25.86
	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	690	6004	173	1688	11319	5047	20339	3229	175	1.37
	$F^{DWB}_{\underline{cq},\overline{cq}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	625	8586	155	2249	7264	5766	13556	572	363	1.61
15	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	-
	$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	-
	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	649	5810	157	2295	15694	7317	19256	4133	254	1.74
	$F^{DWB}_{\underline{cq},\overline{cq}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	610	8424	155	2249	9069	7256	14286	604	504	2.62
25	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	-
	$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	-
	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	597	5537	137	3214	21744	10978	11674	4768	206	1.96
	$F^{DWB}_{\underline{cq},\overline{cq}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	570	8101	139	3208	12202	10003	13805	560	379	2.22
32	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	-
	$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	-
	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	567	5345	124	3863	24991	12823	9741	4939	230	2.22
	$F^{DWB}_{\underline{cq},\overline{cq}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	522	7523	123	3816	11310	9202	13357	478	505	2.76
48	$F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	-
	$F^{DW}(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$	-	-	-	-	-	-	-	-	-	-
	$F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	-	-	-	-	-	-	-	-	-	-
	$F^{DWB}_{\underline{cq},\overline{cq}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$	548	7334	124	5537	19893	17008	7573	482	414	4.48

Table 6: Final comparative performance for solving the real large-scale instances.

Note that for Dantzig-Wolfe and Benders reformulations we had to adapt the price-and-branch heuristic as it was originally designed for a Dantzig-Wolfe reformulation. The principle is to use a usercut callback in Cplex solver to keep generating cuts during the heuristic step as described in Section 4.5. All benchmark formulations are used to solve real size instances within a total time limit of 8 hours. Table 6 shows results over a set of five 58 nuclear unit stochastic instances corresponding to p58i0-1810sS with S ranging from 1 to 48. Table 6 uses the same entries as Table 5 in Section 5.5, but Gap which is the average gap $\frac{p-b}{b}$ between the integer value p found by the heuristic and the linear relaxation value b. Note that b is the same for all formulations, then it is not useful in the evaluation. Nor it is to show the computation time, as the 8-hour time limit is reached by all solution approaches, but $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ for S=1 in the deterministic case. In the latter case, the average computation time is 4667 seconds to find an optimal solution, whereas it is 1739 seconds to find a primal feasible solution. Finally whenever no integer feasible solution is found within the time limit for at least one of the five instances, this is indicated with symbol "-" in the corresponding cell of Table 6. Similarly whenever an entry in the table is not relevant for a formulation, e.g., # ColGen it in the case of formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ as it is solved directly with Cplex solver or # CutGen it in the case of formulation $F(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$ as there is no Benders cut generation.

The sparse structure of formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ allows Cplex solver to find

solutions with less than 1% to optimality within the 8-hour time limit up to 10 stochastic scenarios, whereas $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ provides good quality solutions in less than 30 minutes in the deterministic case. The time spent solving master problems grows rapidly to more than 6 hours 30 minutes for 10 scenarios, thus explaining why $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ and $F(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{p})$ could not find a good feasible solution within the 8-hour time limit. This calls for the use of column-and-cut generation approaches.

Confirming results from Section 5.5 both $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ and $F^{DWB}_{\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ requires roughly the same number of cut generation iterations and number of cuts to solve the linear relaxation. $F^{DWB}_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ generates once again 30% more columns than $F^{DW}(\boldsymbol{\lambda}, \boldsymbol{\eta})$, thus spending more time to solve the master problems and likewise to solve the LP relaxation for few scenarios. It is worth noting that the time spent solving separation problems and generating cuts represent the largest part of the computation time for few scenarios. Not surprisingly this is the step taking most of the total time when searching for an integer solution with Cplex solver. This is also the reason why none of the formulations using column-and-cut generation does finish within the time limit even for few scenarios. Formulation $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ solved through column-and-cut generation with projection leads to high quality solutions for instances with up to 32 stochastic scenarios. Even though $F^{DWB}_{(\underline{\boldsymbol{cq}},\overline{\boldsymbol{cq}})}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ solves the linear relaxation faster than $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ for more than five scenarios, it does not reflect on the gap as the performance of the heuristic is the other way around. This shows that Cplex solver performance is impaired by the up-sizing of the formulation induced by the additional columns and variables. Beyond 32 scenarios, the LP relaxation solution time using $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ raises up to 7 hours due to the time increase in components SolSep and UpPric by projection, as shown in Section 5.5. Then solving reformulation with cumulative capacity variables via column-and-cut generation becomes the best solution approach. In particular, it allows us to solve the LP relaxation faster and provides us with solutions within 4.48% of optimality.

The presented comparative results lead to define a strategy for solving real instances: for less than 10 aggregated scenarios, use formulation $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$ with Cplex solver whereas for more than 10 scenarios, use $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ instead. Introducing cumulative splitting variable, leading to formulation $F^{DWB}_{\underline{cq},\overline{cq}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$, tackled the projection issue, which is the bottleneck of formulation $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$. In particular, using $F^{DWB}_{\underline{cq},\overline{cq}}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ allows us to find integer solutions up to 48 scenarios, whereas $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ is limited to 32 scenarios. Hence, a perspective for future work would be to find a formulation having both properties of solving the linear relaxation within a time almost constant in the number of scenarios and efficient solution by Cplex solver.

Finally, we estimate the industrial potential savings of the approaches based on the NOPP formulations presented in this article using the EDF Checker. We evaluated the best solution found for each number of aggregated scenarios up to

32. Directly using Cplex solver on $F(\boldsymbol{\delta}, \boldsymbol{s}, \boldsymbol{p})$, the expected savings (computed as in Section 5.4) for ten scenarios over a three year horizon are evaluated to 0.56%. Solving $F^{DWB}(\boldsymbol{\lambda}, \boldsymbol{\eta})$ using the column-and-row generation algorithm followed by price-and-branch heuristic allows us considering 32 aggregated scenarios, thus leading to increased expected savings of 1.11%. To put this in perspective, 1% of gain corresponds to approximately 50 million euros.

6. Conclusion

In this paper we present an efficient network flow-based extended formulation to solve the nuclear outage planning problem of EDF. This formulation comprises two stages. First-stage decisions correspond to nuclear outage planning modeled in networks. This stage is well structured for a Dantzig-Wolfe decomposition by nuclear units. We then investigate different ways of improving the fuel level constraints in each scenario at the second stage. The dispatching of the production is a linear problem, allowing the use of Benders reformulation to decompose the problem by stochastic scenarios.

Computational results performed on real EDF data-set demonstrate the efficiency of the proposed formulation and solution approaches. It is possible to find an optimal solution using an extended arc flow formulation directly by a commercial MIP solver when the number of scenarios is small. However, this method is not sustainable for instances with more than ten scenarios. We resort then to a column-and-row generation approach which gives very high quality solutions to real data-set instances up to 32 scenarios within 8 hours. We identified the bottleneck in this approach to handle more scenarios, as the increase in time to update pricing subproblems and to generate Benders' cuts. Addition of cumulative variables avoids this bottleneck to solve the LP problem; however, the number of columns required to solve the LP relaxation increases and these columns slow down the search of integer feasible solutions. Still, only this method provides integer solutions with a reasonable gap for instances up to 48 scenarios in 8 hours. Finally the economic benefit of increasing the number of scenarios was evaluated through simulations on a large set of scenarios for outage dates resulting from each proposed solution approach.

Characteristics making the NOPP challenging to solve are common in the industry: a large space of first stage decisions linked through non-linear constraints to uncertain second-stage problems. In the field of energy management, unit commitment problems, solved daily, and long-term project scheduling problems also share these characteristics.

Future work might combine the formulation using cumulative variables to solve the LP relaxation with a projected one to get integer solutions. This way one can avoid both the increase of LP relaxation solution time due to projection and the slow down of the MIP solver due to increase number of columns. Initializing the Benders decomposition with rapidly generated cuts might offer computational performance improvements.

Acknowledgements

We thank EDF and the PGMO foundation (https://www.fondation-hadamard.fr/PGMO) for providing support to this work. The computational experiments have been executed on PlaFRIM (https://www.plafrim.fr).

References

[1] Arrow, K. J., Hurwicz, L., and Uzawa, H. (1958). *Studies in linear and non-linear programming*. Stanford Math. Stud. Social Sci. Cambridge Univ. Press.

[2] Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

[3] Birge, J. R. and Louveaux, F. (1997). *Introduction to Stochastic Programming*. Springer-Verlag, New York, NY, USA.

[4] Brandt, F. (2010). Solving a large-scale energy management problem with varied constraints. Master's thesis, Karlsruhe Institute of Technology (KIT), Faculty of Informatics., Germany.

[5] Desrosiers, J. and Lübbecke, M. E. (2011). Branch-price-and-cut algorithms. *Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Chichester*, pages 109–131.

[6] Edwin, K. and Curtius, F. (1990). New maintenance-scheduling method with production cost minimization via integer linear programming. *International journal of Electrical Power & Energy Systems*, 12(3):165–170.

[7] Fortz, B. and Poss, M. (2009). An improved Benders decomposition applied to a multi-layer network design problem. *Operations Research Letters*, 37(5):359–364.

[8] Fourcade, F., Johnson, E., Bara, M., and Cortey-Dumont, P. (1997). Optimizing nuclear power plant refueling with mixed-integer programming. *European journal of operational research*, 97(2):269–280.

[9] Gavranović, H. and Buljubašić, M. (2013). A hybrid approach combining local search and constraint programming for a large scale energy management problem. *RAIRO - Operations Research*, 47(4):481–500.

[10] Godskesen, S., Jensen, T. S., Kjeldsen, N., and Larsen, R. (2013). Solving a real-life, large-scale energy management problem. *Journal of Scheduling*, 16(6):567–583.

[11] Joncour, C. (2011). *2D-orthogonal packing and scheduling problems: modelling by graph theory and mathematical programming approaches*. PhD thesis, Université Sciences et Technologies - Bordeaux I.

[12] Jost, V. and Savourey, D. (2013). A 0–1 integer linear programming approach to schedule outages of nuclear power plants. *Journal of Scheduling*, 16(6):551–566.

[13] Lusby, R. M., Muller, L. F., and Petersen, B. (2010). A solution approach to the roadef/euro 2010 challenge based on benders' decomposition. Technical report.

[14] Mukerji, R., Merrill, H. M., Erickson, B. W., Parker, J., and Friedman, R. (1991). Power plant maintenance scheduling: optimizing economics and reliability. *IEEE Transactions on Power Systems*, 6(2):476–483.

[15] Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2018). Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360.

[16] Porcheron, M., Gorge, A., Juan, O., Simovic, T., and Dereu, G. (2010). *Challenge ROADEF/EURO 2010: A large-scale energy management problem with varied constraints*.

[17] Rozenknop, A., Calvo, R. W., Alfandari, L., Chemla, D., and Létocart, L. (2013). Solving the electricity production planning problem by a column generation based heuristic. *journal of Scheduling*, 16(6):585–604.

[18] Sadykov, R., Vanderbeck, F., Pessoa, A., Tahiri, I., and Uchoa, E. (2019). Primal heuristics for branch and price: The assets of diving methods. *INFORMS Journal on Computing*, 31(2):251–267.

[19] Slyke, R. M. V. and Wets, R. (1969). L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663.

[20] Vanderbeck, F. (2011). Branching in branch-and-price: a generic scheme. *Mathematical Programming*, 130(2):249–294.