Real-world hydro-power unit-commitment: Dealing with numerical errors and feasibility issues

Youcef Sahraoui, Pascale Bendotti, Claudia d'Ambrosio

Real-world Hydro-power Unit-Commitment: Dealing with Numerical errors and Feasibility issues

Youcef Sahraoui[a,b], Pascale Bendotti[a], Claudia D'Ambrosio*[,b]

[a] *EDF R&D, Département OSIRIS, 92141 Clamart, France*
e-mail: pascale.bendotti@edf.fr
[b] *LIX CNRS (UMR7161), École Polytechnique, 91128 Palaiseau Cedex, France*
e-mail: {sahraoui,dambrosio}@lix.polytechnique.fr

Abstract

This article deals with feasibility issues of the hydro-unit commitment relative to units along a valley in the price-taker revenue-maximizing setting. The problem is formulated as a mixed-integer linear programming model. Besides physical constraints, we consider two additional specifications that apply to a subset of units and reservoirs within a valley, namely the power-flow curves of each unit feature discrete operational points and each reservoir level should meet target volumes. These specifications, together with the standard issues affecting real-world data, make our problem harder to solve, often infeasible. We follow a step-by-step approach to identify and repair one source of infeasibility at a time, namely numerical errors and model infeasibilities. The former is analyzed and fixed through tools like an exact solver and a model and data preprocessing. The remaining infeasibilities are eliminated with a 2-stage method. In the first stage, a minimal deviation from target volumes, i.e., strategic, thus relaxable, constraints, is computed to make the problem feasible. In the second stage, the original problem is solved with a possible deviation from the target volumes as defined in the first stage. Computational results confirm the effectiveness of the proposed method to recover feasibility on a challenging real-world test set.

Keywords: integer linear programming, Hydro Unit-Commitment, numerical errors, feasibility

1. Introduction

At a large scale, electricity is a commodity that can hardly be stored; this implies that an electricity producer cannot sit on an inventory of past excess production to meet the demand for its consumers at every time.

To satisfy the production-demand equilibrium, planning ahead the management of production is required, ranging from long-run facility investments to intra-daily rescheduling. The usual paradigm is to make strategic decisions for the long-term horizon, tactical planning for the mid-term horizon, and operational scheduling for the short-term horizon. At the short-term horizon, for all units, the stake is to schedule which unit to commit such that the overall demand is met and that units' operational constraints are satisfied. This scheduling problem is referred to as the *unit-commitment* problem (UCP).

At Électricité de France (EDF), main means of production are thermal power plants (both nuclear and fossil-fuel) and hydro-power plants. Other means of production come from wind power, solar energy, bio energy. While thermal power plants somewhat operate independently from one another, hydro-power plants are always considered within *valleys*; a valley is a network of interconnected reservoirs and multi-unit pump-storage hydro-power plants. Then a power plant consists of one or several units. At the short-term horizon, information on forecasted demand, unit availability, fuel costs, etc. is assumed accurate and we consider a deterministic setting. EDF being independent with respect to the Transmission System Operator, transmission constraints are not considered.

For the EDF large-scale UCP (see [1] for more details), the solution method is a price decomposition based on a Lagrangian Relaxation (LR). The main idea is to take advantage of the special structure of the UCP, where all thermal power plants and valleys are coupled through demand. Basically, coupling constraints weighted by Lagrangian multipliers are moved to the objective function, such that the relaxed problem decomposes into independent sub-problems; Lagrangian multipliers are updated using a subgradient method [2] or a bun-

dle method [3]. Each valley (resp. each thermal power plant) is considered as a sub-problem and is solved using mixed integer programming (resp. dynamic programming), as presented in [4]. In an attempt to recover primal feasibility, the Augmented Lagrangian method as an extension to the standard LR approach is used as a second stage. The principle is to add a quadratic penalization of the relaxed constraints alongside the linear weight of standard LR to the objective function. Since the quadratic term compromises the decomposability property, a partial linearization as in [5] is used.

Short-term scheduling of hydro-power plants within a valley is referred as the *hydro unit-commitment* (HUC) problem (see [6, 7, 8] for surveys). It arises for example for an independent electricity producer managing a single valley. In this work, we focus on the valley sub-problem of the LR decomposition scheme where the constraint on demand has been relaxed and the prices in the objective function are Lagrangian multipliers. More precisely, we consider a deterministic HUC problem for a revenue-maximizing price-taker scheduler, where power prices, water usage values, and reservoirs' external inflows are given parameters. The objective of this problem is to maximize power revenues and value of water remaining at the end of the planning horizon. In this model, power output only depends on water flow and does not depend on reservoirs' water levels, i.e., *head and tailrace effects* are ignored, and power-flow curves feature discrete points and affine segments.

1.1. Unit-commitment: state of the art

In a broader perspective, the UCP has been subject to a large research activity due to its practical importance (see surveys [9, 10] and more recently [11]). However, it can still not be considered as a well-solved problem. To be more specific, we focus on the HUC sub-problem of the UCP in a deterministic setting. It involves a time horizon ranging from one to a couple of days, represented by a discrete set of periods ranging from 24 one-hour periods to a few hundred 5-minute periods, and needs to be solved in a very limited time frame. The UCP is a large-scale, non-convex optimization problem. For hydro-power

plants, non-convexities are induced by, for example, power-flow curves.

Dynamic programming (DP) is one of the classical approaches for solving sub-problems in a decomposition scheme. More specifically for the HUC problem, a DP scheme is proposed in [12] for an optimal dispatch of turbine units of the Itaipu hydro-power plant; the hydro-power plant considered is composed of several parallel turbine units. The results emphasize the importance of considering individually each turbine unit and its start-up/shut-down. Indeed, depending on the choice of start-up/shut-down costs, the trade-off solution can be a schedule with numerous start-up/shut-downs with low power losses or a flatter but less efficient generation schedule.

Mixed Integer Linear Programming (MILP) is another well-accepted modeling and solving technique used for the HUC problem (see references in surveys [11, 8]), especially since MILP solvers have proven their efficiency. The nice features for using it are that constraints can be easily added and that non-linearities can be approximated as piecewise linear functions. We refer the reader to [13, 14] for a detailed approximation of the head effect for example. Furthermore, the combinatorial aspects of the problem can be modeled using discrete variables. However, the inherent complexity of the solution technique provides no guarantee in terms of time it takes to reach optimality. Even worse, finding a feasible solution could be just as difficult and could even be impossible in a given time limit. As reported in [15], most existing MILP models fail when trying to solve directly a UCP without an initial feasible solution. Models for unit commitment involve multiple constraints with intricate variables, therefore it is not always trivial to know whether a problem is feasible or infeasible *a priori*. Proving or disproving the feasibility of problems can require a lot of computational efforts. The reason is that the branch-and-bound algorithm, i.e., the classical method used to solve MILP problems, is based on dividing the MILP in subproblems. The principle is to branch and explore the subproblems which continuous relaxation is feasible and could potentially contain an integer feasible solution better than the best found so far. Thus, some subproblems could be eliminated thanks to the current best feasible solution, but this is not

possible when the problem is infeasible as no solution could be found. Consequently, this might end up in exploring a larger feasible space than for the case of a feasible instance. The status of a problem instance can remain undetermined when solution methods fail to terminate. For example in [16], the authors derive a necessary and sufficient condition on the feasibility of HUC solutions in the case of a single hydro-power plant with one reservoir. Checking this condition allows to avoid unnecessary computations in case of infeasibility and also to build a feasible schedule in case of feasibility. This condition is used within a branch-and-bound scheme to implicitly enumerate feasible schedules or to prove no feasible one exists. Several units, each with several operating states, fed by a single reservoir are considered in this HUC problem. The method cannot however be directly extended to the HUC problem we study as dynamic constraints such as ramping rates and multi-reservoir interacting flows are not taken into account.

1.2. Problem statement

In this work, we consider an MILP approach to solve the price-taker revenue-maximizing HUC problem as proposed in [4] and we focus on feasibility issues. Firstly, the solution obtained through computations can be affected by numerical errors. Indeed, floating-point representation of large (volumes) and small (flows) variables introduces rounding errors. Because reservoirs levels can be regarded as integrators with respect to flows, such errors are amplified and eventually lead to inaccurate feasibility results. This is confirmed by the condition number of the constraint matrix as we will discuss in the following. In this sense the HUC problem is expected to be intrinsically sensitive to numerical errors. Secondly, we try to solve real-world instances, whose consistency needs to be checked because data happens to be inaccurate. Thirdly, on top of featuring multiple reservoirs, our model has two additional specifications that are induced by discrete operational constraints and by water-management policies. Indeed, the first aforementioned specification requires that the schedule for a subset of units along a valley should be defined over discrete operational points, while

the second specification requires that each reservoir level along a valley should meet mid-horizon and final target volumes. In practice, these conflicting specifications cause infeasibility situations. Note that head and tailrace effects are assumed negligible as water levels in upstream and downsteam reservoirs are almost invariant over the planning horizon. Other power efficiency losses that are not due to water levels are considered through a piecewise linear approximation (see Section 2.2.2).

As far as we know, most existing work on the HUC either deals with different model features and/or presumes of feasibility and numerical exactness. In this work, infeasibilities indicate that consistency of data and model needs to be reviewed so that the problem admits a solution. We analyze precisely the sources of infeasibilities. Our contribution is to propose a methodology to classify feasibility issues and a preliminary processing of both model and data to obtain solutions free from numerical errors and appropriate for the real-world problem.

A step-by-step approach is taken. The idea is to provide a method to isolate the practical difficulties — rounding errors, data errors, data inconsistency, model infeasibilities — to cope with them one at a time. Firstly, we present a complete model, which incorporates all operational specifications, and illustrate through experiments a few of the aforementioned practical difficulties. Secondly, we derive a simple model to analyze numerical errors; the simple model preserves the basic continuous characteristics of a standard HUC problem, e.g. conservation of water and bounds on reservoirs, but removes the discrete operational requirements. As we focus on feasibility issues, the key idea is to design the simple model so that its feasible set contains the feasible set of the complete model. An exact solver is used to check consistency of the results obtained using a floating-point solver and to detect rounding errors. The effects of data errors on feasibility is mitigated by the introduction of marginal corrective slacks in the model. Once numerical issues are dealt with, consistency of model and data is checked with respect to both additional specifications - discrete operations and target volumes. The derivation of several relaxations of the complete model al-

lows us to analyze and classify infeasibilities. Finally, while data inconsistencies are discarded, model infeasibilities due to target volumes are eliminated with a 2-stage method. In the first stage, a minimal deviation from target volumes is computed to make the problem feasible. In the second stage, the original HUC problem is solved with a possible deviation from the target volumes as defined in the first stage.

This article is organized as follows. Section 2 is devoted to the description of the complete model for the considered HUC problem. In Section 3, first computational tests are performed using an MILP solver as a black box on a test set of real-world instances, thus illustrating a high occurrence of infeasibilities. The simple model is derived in Section 4 and numerical errors are analyzed by resorting to an exact solver. In Section 5, an analysis of the different sources of infeasibilities is carried out and a 2-stage method with feasibility recovery is proposed. Final experimental results show its effectiveness on the original test set considered. Some concluding remarks and future directions end the article.

2. Model description

We now present in details the *complete* mathematical model starting by introducing some notations. It is denoted *complete* as opposed to the *simple* one introduced in Section 4.1.

2.1. Notations

In this section, we present sets and indices, parameters, and variables, with International System base units. For proprietary reason, we give neither real units as used in the data nor orders of magnitude of the parameter values though unit choice is vital for numerical precision (see Section 4.2.1 for more details).

2.1.1. Sets and indices

$T = \{1, \ldots, \bar{t}\}$ set of time periods.

For simplicity, we assume that the number of time periods $\bar{t}$ is even.

R set of reservoirs.

$R^C \subset R$ set of reservoirs without storage capacity.

$R^T \subset R$ set of reservoirs managed with target values and water usage value.

$\{R^C, R^T\}$ is a partition of R. Reservoirs in R^C represent structures that contain water but that cannot store it properly like very small basins or canals; such reservoirs are found upstream of run-of-the-river plants where all incoming inflow must be discharged; such structures are still referred to as reservoirs for the sake of simplicity though it is somewhat improper with respect to actual reservoirs in R^T.

I set of units.

$I^C \subset I$ set of units with continuous operations.

$I^D \subset I$ set of units with discrete operations.

$I^G \subset I$ set of generating units.

$I^P \subset I$ set of pumping units.

$\{I^C, I^D\}$ is a partition on I, and $\{I^G, I^P\}$ is also a partition of I. Note that the generating units release water to produce power while pumping units consume power to pump water. The convention chosen here is to model power with algebraic values (positive or negative) and flow with absolute values along with a sign that depends on the direction of the flow. It is worth noting that when a unit i is located downstream of a reservoir $r \in R^C$ with no storage capacity, the unit should allow for instantaneous response for an instance of this problem to be feasible. Basically, the upstream flow relative to unit $i \in I^C$ at each period is equal to its downstream flow. This would be the case for run of the rivers plants. This explain why units I^C are with continuous operations.

$I^R \subset I^G \times I^P$ set of pump-storage stations.

Each station is represented by a couple (i', i'') with generating unit i' and pumping unit i'' such that both units are between the same upstream reservoir and the same downstream reservoir.

J_{ti} set of operational points of unit i at time period t $(\forall t \in T, \forall i \in I)$. We assume w.l.o.g. that $|J_{ti}| \geq 1$.

The set of possible operational points may vary in time depending on turbines availability, for example.

$R_i^+, R_i^- \in R$ upstream reservoir, downstream reservoir of unit i, respectively $(\forall i \in I)$.

$I_r^+, I_r^- \subset I$ sets of upstream units, downstream units of reservoir r, respectively $(\forall r \in R)$.

2.1.2. Parameters

Π period duration [s].

Ω_i time lapse for water to flow through unit $i \in I^G$ to downstream reservoir R_i^- once water is released from upstream reservoir R_i^+ or to be pumped up through unit $i \in I^P$ from downstream reservoir R_i^- to upstream reservoir R_i^+ $(\forall i \in I)$ [expressed in number of time periods].

Ψ_{0r} initial volume in reservoir r $(\forall r \in R^T)$ [m^3].

$\overline{\Psi}_r, \underline{\Psi}_r$ maximum, minimum volume capacity in reservoir r, respectively $(\forall r \in R^T)$ [m^3].

$\overline{A}_r, \underline{A}_r$ mid-horizon maximum, mid-horizon minimum target volumes in reservoir r, respectively $(\forall r \in R^T)$ [m^3].

$\overline{B}_r, \underline{B}_r$ final maximum, final minimum target volumes in reservoir r, respectively $(\forall r \in R^T)$ [m^3].

Γ_{tr} external water inflow in reservoir r during period t, that is to say available at end of period t $(\forall t \in T, \forall r \in R)$ [m^3/s].

Inflows can be positive for precipitations or runoff water; inflows are negative for evaporation or water extraction.

$\overline{\Upsilon}_i, \underline{\Upsilon}_i$ maximum water flow ramp-up, maximum water flow ramp-down between two consecutive time periods at unit i, respectively ($\forall i \in I$) [m^3/s].

Φ_{tij} water flow increment released at time period t, through unit i, corresponding to operational point j ($\forall t \in T, \forall i \in I, \forall j \in J_{ti}, \Phi_{tij} > 0$) [$m^3$/s].

$\overline{\Sigma}_{ti}$ maximum spillage allowed during time period t through unit i ($\forall t \in T, \forall i \in I$) [$m^3$/s].

To have a consistent notation, spillage is artificially introduced for pumps and $\forall t \in T, \forall i \in I^P, \overline{\Sigma}_{ti} = 0$.

Λ_{tij} power generated at time period t by unit i corresponding to operational point j ($\forall t \in T, \forall i \in I, \forall j \in J_{ti}$) [W].

ω_r water value of reservoir r ($\forall r \in R^T$) [currency/m^3].

Water value is assumed constant with respect to volume as water volumes in reservoirs are almost invariant over the planning horizon.

λ_t proportional price for power at time period t ($\forall t \in T$) [currency/W per time period].

$\overline{Q}_{ti}$ maximum water flow released (or pumped) at time period t, through unit i ($\forall t \in T, \forall i \in I$) [$m^3$/s].
$\overline{Q}_{ti} = \sum_{j \in J_{ti}} \Phi_{tij}$

$\underline{Q}_{ti}$ minimum water flow released at time period t, through unit i ($\forall t \in T, \forall i \in I$) [$m^3$/s].

$$\underline{Q}_{ti} = \begin{cases} \Phi_{ti1} & \text{when unit } i\text{must operate at } t \\ 0 & \text{otherwise} \end{cases}$$

$\overline{P}_{ti}$ maximum power generated during time period t by generating unit i ($\forall t \in T, \forall i \in I^G$) [W].
$\overline{P}_{ti} = \max_{k \in J_{ti}} \sum_{j=1}^{k} \Lambda_{tij}$

$\underline{P}_{ti}$ its absolute value corresponds to the maximum power consumed during time period t by pumping unit i ($\forall t \in T, \forall i \in I^P$) [W].
$\underline{P}_{ti} = \min_{k \in J_{ti}} \sum_{j=1}^{k} \Lambda_{tij}$

2.1.3. Variables

v_{tr} water volume in reservoir r at end of time period t ($\forall t \in T, \forall r \in R$) [$m^3$].

q_{ti} water flow released through unit i during time period t ($\forall t \in T, \forall i \in I$) [$m^3/s$].

s_{ti} water flow spilled through unit i during time period t ($\forall i \in I, \forall t \in T$) [$m^3/s$].

p_{ti} power generated by unit i during time period t ($\forall t \in T, \forall i \in I$) [W].

x_{tij} binary activation status of discrete operational point j of unit i during time period t ($\forall t \in T, \forall i \in I, \forall j \in J_{ti}$).

y_{tij} continuous activation status of operational point j of continuous-operating unit i during time period t ($\forall t \in T, \forall i \in I^C, \forall j \in J_{ti}$)

z_{ti} binary variable allowing spillage at unit i in time period t ($\forall t \in T, \forall i \in I$).

To simplify notation, variables are not defined for initial values (for $t \leq 0$) but they may be introduced when needed with specific values.

2.2. Complete mathematical model

This section is devoted to the description of the complete mathematical model. We start with bounds on variables, continue with the constraints, and finally introduce the objective function. By convention, sums on empty sets are equal to zero.

2.2.1. Variable bounds

Now we introduce the simple bounds on variables introduced in the previous section ($\forall t \in T$):

$$\underline{\Psi}_r \leq v_{tr} \leq \overline{\Psi}_r \qquad \forall r \in R \tag{1}$$

$$0 \leq \underline{Q}_{ti} \leq q_{ti} \leq \overline{Q}_{ti} \qquad \forall i \in I \tag{2}$$

$$0 \leq s_{ti} \leq \overline{\Sigma}_{ti} \qquad \forall i \in I \tag{3}$$

$$0 \leq p_{ti} \leq \overline{P}_{ti} \qquad \forall i \in I^G \tag{4}$$

$$\underline{P}_{ti} \leq p_{ti} \leq 0 \qquad \forall i \in I^P \tag{5}$$

$$x_{tij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J_{ti} \tag{6}$$

$$y_{tij} \in [0,1] \qquad \forall i \in I^C, \forall j \in J_{ti} \tag{7}$$

$$z_{ti} \in \{0,1\} \qquad \forall i \in I. \tag{8}$$

Note that valves can be modeled with $\overline{\Sigma}_{ti} > 0$ and $J_{ti} = \emptyset$.

2.2.2. Constraints

We start by the constraints modeling the water volume conservation at time period t and reservoir r ($\forall t \in T, \forall r \in R_T$):

$$\begin{aligned} v_{tr} = v_{(t-1)r} &+ \Pi\Gamma_{tr} \\ &+ \Pi \sum_{\substack{i \in I_r^+ \cap I^G \\ t-\Omega_i \geq 1}} (q_{(t-\Omega_i)i} + s_{(t-\Omega_i)i}) \\ &+ \Pi \sum_{\substack{i \in I_r^- \cap I^P \\ t-\Omega_i \geq 1}} (q_{(t-\Omega_i)i} + s_{(t-\Omega_i)i}) \\ &- \Pi \sum_{i \in I_r^- \cap I^G} (q_{ti} + s_{ti}) \\ &- \Pi \sum_{i \in I_r^+ \cap I^P} (q_{ti} + s_{ti}) \end{aligned} \tag{9}$$

where v_{0r} and $(q_{ti}, s_{ti})_{t \leq 0}$ are replaced with their initial values.

The water volume in the reservoir is equal to the water volume at the previous time period plus the external forecasted inflows released from the upstream

units (first summation), and pumped from the downstream units (second summation), minus the outflows released by the downstream units (third summation) and pumped by the upstream units (fourth summation). The constraints take into account the time lapse needed for water to reservoir r.

For each reservoir r without storage capacity ($\forall r \in R^C$), the water volume conservation constraint at each time period t ($\forall t \in T$) is slightly different:

$$\begin{aligned}\Gamma_{tr} &+ \sum_{\substack{i \in I_r^+ \cap I^G \\ t-\Omega_i \geq 1}} (q_{(t-\Omega_i)i} + s_{(t-\Omega_i)i}) \\ &+ \sum_{\substack{i \in I_r^- \cap I^P \\ t-\Omega_i \geq 1}} (q_{(t-\Omega_i)i} + s_{(t-\Omega_i)i}) \\ &- \sum_{i \in I_r^- \cap I^G} (q_{ti} + s_{ti}) \\ &- \sum_{i \in I_r^+ \cap I^P} (q_{ti} + s_{ti}) = 0. \end{aligned} \tag{10}$$

Like above without the volume variables, water is conserved: outgoing flows balance incoming flows, except that water cannot be stored.

On top of the reservoir capacity (1), mid-horizon and final volumes of reservoirs $r \in R^T$ are subject to targets:

$$\underline{A}_r \leq v_{\frac{\bar{t}}{2}r} \leq \overline{A}_r \qquad \forall r \in R^T \tag{11}$$

$$\underline{B}_r \leq v_{\bar{t}r} \leq \overline{B}_r \qquad \forall r \in R^T. \tag{12}$$

Target bound values are indicators provided by mid-term planning to avoid being blind to the future in the short-term scheduling. They are border stones controlling the trajectories of water levels , but they are strategic constraints, thus relaxable, and not physical constraints, thus unrelaxable, as the others.

We now introduce a system of constraints to express power-flow curves for units with continuous operations. Constraints (13), (14), (15), and (16) describe piecewise linear power-flow curves where the breakpoints are the operational points; the piecewise linear curves are expressed with the *incremental*

formulation, see [17], i.e., $\forall t \in T, \forall i \in I^C$,

$$q_{ti} = \sum_{j \in J_{ti}} y_{tij} \Phi_{tij} \tag{13}$$

$$p_{ti} = \sum_{j \in J_{ti}} y_{tij} \Lambda_{tij} \tag{14}$$

$$x_{tij} \leq y_{tij} \qquad \forall j \in J_{ti} \tag{15}$$

$$y_{ti(j+1)} \leq x_{tij} \qquad \forall j \in \{1, \ldots, |J_{ti}| - 1\}. \tag{16}$$

We prefer this formulation over the standard convex-combination formulation as it is known to have nice theoretical and computational properties (e.g., it is locally ideal, see [17] for details). For the discrete-operating units, flow and power can only take a set of discrete values as modeled by constraints (17), (18), and (19); the formulation is a simplified restriction of the incremental formulation in order to express variables that only take a set of discrete values. $\forall t \in T, \forall i \in I^D$,

$$q_{ti} = \sum_{j \in J_{ti}} x_{tij} \Phi_{tij} \tag{17}$$

$$p_{ti} = \sum_{j \in J_{ti}} x_{tij} \Lambda_{tij} \tag{18}$$

$$x_{ti(j+1)} \leq x_{tij} \qquad \forall j \in \{1, \ldots, |J_{ti}| - 1\}. \tag{19}$$

Note that the set of available operational points are chosen by operators according to operating conditions. Note also that, as putting together constraints (17)-(19) with (9), (11), and (12) might lead to infeasibilities for some instances, we will allow in the following "limited" violations of constraints (11) and (12) (see Section 5.2).

We next introduce the set of constraints related to spillage:

$$s_{ti} \leq z_{ti} \overline{\Sigma}_{ti} \qquad \forall t \in T, \forall i \in I^G \tag{20}$$

$$z_{ti} \leq x_{ti(|J_{ti}|)} \qquad \forall t \in T, \forall i \in I^G. \tag{21}$$

Constraints (20) imply that spillage through unit i during time period t can be non-zero only if variable $z_{ti} = 1$. Constraints (21) translate an operational rule

stating that spillage is allowed through each unit only if the last operational point is reached, i.e., $x_{ti(|J_{ti}|)} = 1$. Note that, as no start-up/shut-down cost is considered, the objective function does not discard solutions with positive value for spillage and non-maximal operational points for the same unit and period, thus we need to force this thanks to binary variables z.

Next, we have monotonicity constraints together with disjunctive constraints relative to the simultaneous use of pump and turbine, $\forall t \in T$,

$$1 + x_{(t-1)ij} \geq x_{(t-2)ij^-} + x_{tij^+} \quad \forall i \in I^D, \forall j \in J_{ti} \tag{22}$$

$$x_{(t-1)ij} \leq x_{(t-2)ij^-} + x_{tij^+} \quad \forall i \in I^D, \forall j \in J_{ti} \tag{23}$$

$$x_{ti'1} + x_{ti''1} \leq 1 \quad \forall (i', i'') \in I^R \tag{24}$$

$$x_{ti'1} + x_{(t-1)i''1} \leq 1 \quad \forall (i', i'') \in I^R \tag{25}$$

$$x_{(t-1)i'1} + x_{ti''1} \leq 1 \quad \forall (i', i'') \in I^R, \tag{26}$$

where $(j^-, j, j^+) \in J_{(t-2)i} \times J_{(t-1)i} \times J_{ti}$ are the indices pointing at the same operational point at three consecutive time periods $(t-2, t-1, t)$, *i.e.*, $\sum_{k=1}^{j^-} \Phi_{(t-2)ik} = \sum_{k=1}^{j} \Phi_{(t-1)ik} = \sum_{k=1}^{j^+} \Phi_{tik}$. Also, $x_{0ij}, x_{(-1)ij}$ are replaced with their initial values. Constraints (22)-(23) ensure smooth operations to maintain a sustainable level of stress and wear for the turbine unit within a power plant. If one turbine were to correspond to one operational point, the monotonicity constraints would reduce to 2-period min-up/min-down constraints on operational points instead of minimum uptime and minimum downtime of each unit as studied in [18, 19]. Note that no start-up costs have been considered in this model. Instead, start-ups/shut-downs over two consecutive periods are prohibited through constraints. Constraints (24)-(26) impose that the pump and turbine of a reversible station cannot operate at the same time and consecutive operations require a one-period transition.

Finally, we present the ramp-up and ramp-down constraints, $\forall t \in T, \forall i \in I$,

$$(q_{ti} + s_{ti}) - (q_{(t-1)i} + s_{(t-1)i}) \leq \overline{\Upsilon}_i \tag{27}$$

$$(q_{ti} + s_{ti}) - (q_{(t-1)i} + s_{(t-1)i}) \leq -\underline{\Upsilon}_i \tag{28}$$

where q_{0i}, s_{0i} are replaced with their initial values. Like monotonicity constraints (22)-(23), ramp-up and ramp-down constraints protect excessive tear and wear of turbines, especially as no start-up/shut-down costs are considered. Also, ramp-up and ramp-down constraints prevent large drift changes in flows that could have harmful downstream effects. The restriction on flow changes dependent on the use of water downstream, for example safety reasons whenever the water flow involves the level of water in rivers used for leisure activities as, for example, fishing or kayaking.

2.2.3. Objective function

Our aim is to maximize the revenues given as:

$$\varrho = \max \sum_{t \in T} \sum_{i \in I} \lambda_t p_{ti} + \sum_{r \in R^T} \omega_r (v_{\bar{t}r} - \Psi_{0r} - \Pi \sum_{t \in T} \Gamma_{tr}). \tag{29}$$

The first summation gives the profit/cost due to power generation/consumption during the planning horizon (recall that $\boldsymbol{\lambda}$ is the proportional price vector), while the second summation gives the net value of the water volume remaining at the end of the planning horizon for reservoirs $r \in R^T$. Prices can originate from market forecasts or they can also correspond to Lagrangian multipliers. Water values are dummy opportunity costs: they reflect the value of the optimal use of water in the future instead of using it now. They are indicators provided by mid-term planners. Different ways to derive water values that take into account uncertainty of water inflows to manage reservoirs in the mid-term are presented in [20] for example.

3. First computational tests

In this section, we present characteristics of the test set and first computational results with the complete model presented in Section 2.

3.1. Test set and configuration

Our test set is composed of 66 real-world instances corresponding to all combinations of 6 valleys at 11 dates. For all instances, $\bar{t} = 96$ time periods

are considered. Table 1 summarizes instance characteristics per valley. First column gives valley name and other entries are:

$|R|$ number of reservoirs

$|I|$ number of units

$|I^R|$ number of pump-storage stations

$|I^C|$ number of continuously-operating units

#bin vars average number of binary variables per valley over the 11 dates and standard deviation in parenthesis

#vars average number of variables per valley over the 11 dates and standard deviation in parenthesis.

Valley topographies - order and structure of reservoirs and units along streams - remain unchanged across instances of each valley. Operating conditions vary from one date to another, therefore data such as initial values, volume bounds, water inflows, water values, electricity prices, sets of operational points vary. The number of variables changes as well as the sets of operational points change over the time horizon and from one date to another. The variations in valley topography and date for our test set were chosen to be representative of possible configurations. Since we have a compact formulation where binary variables correspond directly to operational points, the average number of binary variables is somewhat an indicator of the combinatorial difficulty of the instances.

Tests were executed on 64-bits Intel Xeon CPU E5504 running at 2.00 GHz x8 cores with Linux and 11.7 GB of RAM. The mathematical programming modeling language AMPL Version 20121017 was used to run the MILP solver IBM ILOG CPLEX 12.4.0.0, see [21]. For each instance, a time limit of 1200 seconds was imposed. Tolerance parameters were left to their default values: (linear) feasibility $\epsilon_f = 10^-6$, integer feasibility $\epsilon_i = 10^-5$, (linear) optimality $\epsilon_o = 10^-6$.

3.2. Test results

Table 2 summarizes the solution obtained by CPLEX for the 66 instances presented in Section 3.1 with the complete model described in Section 2. CPLEX solution status of an MILP instance can be *integer infeasible* when the instance is proved infeasible, *feasible* when a feasible solution is found, or *intractable* when no integer solution is found within time limit, leaving feasibility of the instance an open question.
Table entries are:

v valley label

time average wall-clock solution time and standard deviation in seconds.

#nodes average number of branch-and-bound nodes and standard deviation in thousands

#inst feasible number of feasible instances

gap (%) average relative gap and standard deviation (only for feasible instances)

$\|\varrho\|$ **(%)** average normalized revenues and standard deviation (as derived in (30) and only for feasible instances)

#inst-no sol. number of instances with no solution as solution is found integer infeasible or intractable within the time limit.

For proprietary reasons, we do not show explicitly values of revenues and deviations. For indication, we show normalized values:

$$\|\varrho\| = \begin{cases} \frac{\varrho - \underline{\varrho}}{\overline{\varrho} - \underline{\varrho}} & if\ \overline{\varrho} > \underline{\varrho} \\ 0 & if\ \overline{\varrho} = \underline{\varrho} \end{cases} \tag{30}$$

where ϱ is the revenues given by the best solution found for a specific instance while $\overline{\varrho}$ and $\underline{\varrho}$ are the maximum and minimum revenues found as best solution values of all instances. The norm of deviations $\|\varepsilon\|$ is computed by normalizing by the maximum sum of deviations ε over all instances.

In addition to the numerous infeasible and intractable cases, several messages from the solver indicate numerical problems: bad average condition numbers and rounded integer variables. Numerical problems , such as rounding errors, data errors, data inconsistency, and model infeasibilities, may indicate inaccurate solutions.

When working with this real-world optimization problem, we run into specific practical difficulties, namely numerical problems and infeasibilities. Firstly we must ensure numerically-computed results are affected neither by the errors inherent to floating-point computations nor by noisy data from real-world instances. Then, since we are considering a real problem that we want to model so as to be feasible, we must update our model and/or adjust our data so as to recover feasibility.

4. Dealing with numerical errors

A numerical error is observed at the end of a computation when a numerically-computed result is different from the exact solution.

For an optimization problem, a major kind of error we can run into is when feasibility is altered. Two cases may happen: either computations find a problem infeasible while exact feasible solutions exist, or computations provide a solution said to be feasible while the exact feasible set is empty. This kind of errors is denoted as *feasibility inconsistency*. In this work, given the large occurrence of infeasibilities, as shown in Table 2, we will focus on identifying and mitigating feasibility inconsistencies.

Most standard MILP solvers - like Cplex - are based on floating-point arithmetic; such arithmetic allows fast computations but inherently incurs rounding errors, which may lead to feasibility inconsistency. The use of an exact solver based on rational arithmetic should guarantee the exactness of our solution (see Section 4.2.2 below for more details). Unfortunately, exact solvers are too slow to get results within satisfactory time on large-scale instances.

In Section 4.1, we introduce a simple version of the model presented in

Section 2 to work with a model tractable with both types of solvers and avoid undecidable situations where no solution is found within time limit.

Once feasibility consistency is dealt with: either an instance is known to be exactly infeasible or an instance is known to be exactly feasible. In the former case, infeasibility becomes a cause for modeling concern since we are dealing with a real-world problem that should be modeled so as to be feasible (see subsequent Section 5). In the latter case, numerical errors related to the feasibility (optimality) certificate of the computed feasible (optimal) solution must be checked. For linear programs, they respectively correspond to satisfaction of linear constraints and non-negativity of reduced costs - when minimizing. For mixed-integer programs, errors on feasibility certificate can be observed by checking integrality of integer variables on top of satisfaction of linear constraints; errors on optimality are less trivial to check. These kinds of errors are not studied in this work.

4.1. Mathematical formulation of a simple model

This section is devoted to the description of a simple version of the model presented in Section 2.2. The idea is to work with a more tractable model and avoid undecidable situations where no solution is found within satisfactory time.

Assuming the solution process is slow mainly because of the combinatorial difficulties in the complete model, no binary variable is considered in the simple model. Instead of considering a standard linear relaxation of the complete model, we prefer to remove all constraints involving binary variables, including the disjunctive constraints (22)-(23) relative to the simultaneous use of pump and turbine.

Following same notation as in Section 2, the mathematical formulation of the simple model is as follows:

- bounds

$$(1)$$

$$0 \leq q_{ti} \leq \overline{Q}_{ti} + \overline{\Sigma}_{ti} \qquad \forall t \in T, \forall i \in I \tag{31}$$

$$(4) - (5)$$

- constraints

$$(9) - (12)$$

$$p_{ti} = \frac{\sum_{j \in J_{ij}} \Lambda_{tij}}{\overline{Q}_{ti} + \overline{\Sigma}_{ti}} q_{ti} \qquad \forall t \in T, \forall i \in I \tag{32}$$

$$(27) - (28)$$

- objective

$$(29)$$

In the simple model, we get rid of spillage (no variables s_{ti} are considered) and aggregate it with the water flow whose domain is extended accordingly, see (31). Simply put, the power curve (32) is a linear segment between $(0, 0)$ and the last operational point with maximum spillage $(\overline{Q}_{ti} + \overline{\Sigma}_{ti}, \sum_{j \in J_{ij}} \Lambda_{tij})$.

For a given instance, the feasibility set of the simple model strictly includes the feasibility set of the complete model since constraints (17)-(26) involving binary variables are removed. Therefore less infeasibilities will happen with the simple model. As for the objective function, the simple model is not strictly speaking a relaxation of the complete one because the expression of the power-flow curve (18)-(14) is given by constraint (32). Note that the power variables p_{ti} are dependent variables which appear only in the objective function.

All in all, the simple model enables us to get rid of intractabilities, and a few infeasibilities, to focus on numerical errors.

4.2. Dealing with floating-point rounding errors

As defined in the IEEE Standard for Floating-Point Arithmetic (IEEE 754), the binary double-precision format is a floating-point system in base 2 with a

52-bit precision. It is a commonly-used format for numerical computations. The set of possible representations in that format is noted $\mathbb{F}$. A number $a \in \mathbb{R}^*$ is represented by $fl(a) \in \mathbb{F}$ such that:

$$fl(a) = (-1)^{s(a)} \cdot m(a) \cdot 2^{e(a)-53} \tag{33}$$

where sign $s(a) \in \{0, 1\}$ is expressed with 1 bit, significand $m(a) \in \{2^{52}, \ldots, 2^{53}-1\}$ is expressed with 52 bits, exponent $e(a) \in \{-1021, \ldots, 1024\}$ is expressed with 11 bits.

Floating-point representations are not exact, $\mathbb{F} \subsetneq \mathbb{R}$ and it may happen that $fl(a) \neq a$. For any real a such that $|a| \in [2^{-1022}, 2^{1023}]$, unit roundoff $u = 2^{-53} \approx 10^{-16}$ in double precision is an upper bound on the relative error $E_r(fl(a))$ of the approximation of a by $fl(a)$:

$$E_r(fl(a)) = \frac{|a - fl(a)|}{|a|} < u \tag{34}$$

and an equivalent definition is $E_r(fl(a)) = |\gamma| < u$, where $fl(a) = a(1+\gamma)$. Note that the following modified version of the approximation also holds: $fl(a) = \frac{a}{1+\gamma}$, $|\gamma| < u$ (see Theorem 2.2 and 2.3 in [22]). Also absolute error $E_a(fl(a))$ on the floating-point representation is defined as and bounded by:

$$E_a(fl(a)) = |a - fl(a)| < u|fl(a)|. \tag{35}$$

The first rounding error happens when representing a real number in a floating-point format. Other rounding errors occur with calculations. Indeed $\mathbb{F}$ is not closed under the basic arithmetic operations: even if the operands lie in $\mathbb{F}$, the result of such operations may need to be rounded to lie in $\mathbb{F}$. The rounding error on the result of a single operation is also bounded by u. The composition of several operations may propagate small rounding errors into larger errors. The study of such errors incurred by implementations of algorithms is an important subject of numerical analysis, see [22].

4.2.1. Tolerances, scaling, and floating-point-based solvers

As stated in [23], solvers based on floating-point representations, such as Cplex, introduce tolerances in order to deal with the rounding errors that are

inherent to such representations.

To avoid detrimental errors on the feasibility set, linear constraints are marginally relaxed with the introduction of absolute *feasibility tolerance* ϵ_f. For instance, for a variable x and a right-hand-side (RHS) parameter b: while we model a constraint as $x \geq b$, solvers see $x \geq fl(b) - \epsilon_f$. Since RHS terms are indicators of the orders of magnitude of contraints, we will consider rounding errors on RHS terms in our reasoning for the sake of clarity.

Provided $E_a(fl(b)) \leq \epsilon_f$ is satisfied, solvers ensure that a less restrictive constraint $x \geq fl(b) - \epsilon_f$ is considered with floating-point representation:

$$\begin{cases} E_a(fl(b)) \leq \epsilon_f \\ fl(b) \leq b + E_a(fl(b)) \end{cases} \Rightarrow b \geq fl(b) - \epsilon_f.$$

Indeed, it is better to slightly extend the feasible set than to arbitrarily restrict it because of rounding errors.
If we scale down the unit of the left-hand-side (LHS) term, for example:

$$x := x/10 \text{ then } x \geq b \text{ becomes } x \geq b/10$$

and the rounding error on the RHS

$$E_a(fl(b/10)) \approx E_a(fl(b))/10.$$

Of course, an experienced modeler would intuitively select suitable units but we find it useful to state this simple rule of thumb.

When providing an instance to a floating-point-based solver, we should make sure that the following condition holds:

$$E_a(fl(b)) < u|fl(b)| \leq \epsilon_f. \tag{36}$$

Usually the solver's feasibility tolerance is uniquely set across constraints and $\epsilon_f = 10^{-6}$ by default for Cplex. Orders of magnitude should be of the range of 10^{10} so that condition (36) holds, which could be rewritten as:

$$|fl(b)| \leq \epsilon_f/u \approx 10^{10}. \tag{37}$$

Let us consider a simple example limited to some linear constraints to illustrate the complementary effect of scaling to avoid detrimental errors on feasibility:

$$\begin{aligned} x &= 10^{12} y \\ x &\geq 0 \\ y &= fl(c) \end{aligned} \tag{38}$$

where c is a value very close to 0, with $fl(c) = -10^{-16}$. In this case $x = -10^{-4}$ and $|-10^{-4}| > \epsilon_f$ is infeasible. If we scale down the unit of variable x, for example $x := x/10^6$, then equation (38) becomes $x = 10^6 y$. In this case $x = -10^{-10}$ and $|-10^{-10}| < \epsilon_f$, thus is feasible.

In our problem, volumes and related bounds are the numbers with the greatest magnitude. In addition, recursive constraints (9) show that volumes result from several intermediary operations, which often lead to error propagation. Volumes are therefore the most likely to be subject to substantial rounding errors. For this reason, we decide to study the effect of scaling volume-related parameters and variables.

Note that tuning directly the solver's feasibility tolerance could be an option if all constraints' RHS terms were of the same magnitude. As this is not the case, we rather rescale each variable individually so that the constraints in which they are involved are not disturbed by rounding error. In particular, let us consider variables v and q and their respective bounds (1) and (2). The magnitude of their bounds can be very different. Thus, rescaling them individually or not could highly influence the potential rounding error on constraints where both sets of variables appear like (9). This is especially true for instances with $\underline{A}_r = \overline{A}_r$ and/or $\underline{B}_r = \overline{B}_r$.

4.2.2. Computational results

Standard floating-point solvers feature default scaling routines, which are kept activated for the computational tests. Moreover, we find it useful to scale

variables and constraints known to be sensitive, thus rescaling the instance before providing it to the solver.

For volume variables v, we test 5 scalings - denoted with letters from A to E - which correspond to increasing volume units we do not express here for proprietary reasons. Volume-related parameters ($\overline{\Psi}, \underline{\Psi}, \overline{A}, \overline{B}, \underline{A}, \underline{B}, \Gamma, \omega$) and constraints (1), (9), (11), (12) are affected. For example, if we scale volumes by a factor 10, we reformulate ($\forall t \in T, \forall r \in R$):

$$\begin{cases} v_{tr} & \leq \overline{\Psi}_r \\ v_{tr} & = v_{(t-1)r} + \Pi\Gamma_{tr} \dots \end{cases} \text{ into } \begin{cases} v_{tr} & \leq \frac{\overline{\Psi}_r}{10} \\ v_{tr} & = v_{(t-1)r} + \Pi\frac{\Gamma_{tr}}{10} \dots \end{cases}$$

The scaling factor (10 in this example) is computed according to the scaling we want to test and the original units of parameter values. Given volumes' orders of magnitude, the feasibility set, as represented by CPLEX, is expected to be altered when using scalings A and B, because condition (37) is not satisfied.

In order to identify situations where the feasibility status computed by CPLEX is incorrect, CPLEX's results are compared with results of an exact solver. Namely, we use QSOPTEX, an LP solver based on rational arithmetic, instead of floating-point arithmetic [24]. Note that the solution obtained with a rational-based solver is left unchanged by scaling as there is no feasibility tolerance.

Table 3 summarizes feasibility consistency for the CPLEX solution results with QSOPTEX's exact result. The experiment is performed for the 66 instances presented in Section 3.1 with the simple model described in Section 4.1 according to scalings A to E. With regards to feasibility, the CPLEX solution results of a problem formulated with different volume scalings are said to be:

- *consistentF* if instances with all scalings are found feasible in agreement with QSOPTEX's exact result
- *consistentI* if instances with all scalings are found infeasible in agreement with QSOPTEX's exact result
- *inconsistentF* if at least one instance with one scaling is found infeasible in contradiction with QSOPTEX's exact result

- *inconsistentI* if at least one instance with one scaling is found feasible in contradiction with QSoptEx's exact result.

Condition (37) with scaling A and B is not satisfied for volume variables as, decreasing the unit scale from scaling E to A, their corresponding bounds take larger values. The limit defined by inequality (37) is exceeded with scaling A, thus, with it Cplex's solution is wrongfully found infeasible for the 32 *inconsistentF* instances. Note that scaling A and B are characterized by large right-hand-side values, whereas the constraints matrix features a large condition number, i.e., using the 2-norm for matrices, the ratio between the maximum and the minimum singular value of the parameters characterizing the instance. It is well-know that a high value of the condition number corresponds to ill-conditioned instances. An appropriate scaling would limit the effect of input rounding errors on the solution.

Let us now consider the 44 (=12 *consistentF* + 32 *inconsistentF*) instances that are rightfully found feasible and solved to optimality by Cplex for scalings B to E. For each instance, the relative objective error $\%\Delta$ is computed as the relative difference between the objective value found by both solvers. Table 4 summarizes statistics on relative objective error according to scaling. Maximum (max), average (avg), and standard deviation (std) of the relative objective error $\%\Delta$ are computed over the 44-instance set for each scaling.

Even though condition (37) with scaling B is not satisfied, there is hardly no feasibility inconsistency observed for the considered test set. However, the errors on the computed objective value for scaling B appear to be much larger than those obtained with scalings C to E. This indicates that rounding errors are more likely to have an impact on the quality of the solution.

To return to the 4 (=1+1+2) *inconsistentI* instances of Table 3, the results illustrate there is a limit for the use of scalings with Cplex. Indeed, when a relatively large scaling is used, significant digits of handled variables and parameters are neglected as they become lower than the absolute feasibility tolerance. Therefore, the relaxation introduced by the standard feasibility tolerance for

rounding errors becomes too loose and exactly infeasible instances (as seen by QSoptEx) are seen as feasible.

An approximate computation with inaccurate data might recover feasibility by chance, whereas an exact computation with inaccurate data cannot. However, the combination of large scalings and feasibility tolerance for an approximate computation might over-relax the problem. For this reason, scaling E, being too large, is discarded for future computations with CPLEX, while data errors are dealt with in the subsequent section.

4.3. Dealing with data errors

Another kind of numerical error may be observed when inaccurate input data, or *data errors* are involved. As pointed above in Section 4.2.2 (see the *inconsistentI* case), even an exact computation may lead to an incorrect result if data is inaccurate.

If input value $\hat{a}$ is read for parameter a, the absolute error on data $\hat{a}$ is:

$$E_a(\hat{a}) = |\hat{a} - a|.$$

The problem instances were obtained by manipulating raw data. Such manipulation could have introduced errors of various types. We, thus, recovered exact values of a by going back to raw data and checking manually the correctness of the data for some of the instances. This enabled us to derive upper bounds on the relative data error $\overline{E_r}$ by rounding E_r up to the next order of magnitude:

$$\frac{|\hat{a} - a|}{\hat{a}} = E_r(\hat{a}) \lesssim \overline{E_r}(\hat{a})$$

for all parameters:

- $\Psi_0, \overline{\Psi}, \underline{\Psi}, \overline{A}, \overline{B}, \underline{A}, \underline{B}, \Gamma$ regarding reservoirs,

- $\overline{\Upsilon}, \underline{\Upsilon}, \overline{\Sigma}, \Phi, \Lambda$ regarding units.

In the general case since only $\hat{a}$ is given, we assumed the previously derived upper bounds on relative data errors $\overline{E_r}$ were valid to obtain upper bounds on

absolute data errors $\overline{E_a}$:

$$E_a(\hat{a}) \leq \overline{E_a}(\hat{a}) \approx \overline{E_r}(\hat{a}) \cdot |\hat{a}|.$$

Feasible regions are sometimes improperly restricted and found empty because of data errors. To recover feasibilty, we suggest a *data correction* that takes into account upper bounds on data errors to guard against the worst case. More specifically, within each constraint we correct parameters by adding or subtracting the upper bound on the error in the direction that relaxes the constraint. Assuming *w.l.o.g.* $a \geq 0$, $b \geq 0$, $x \geq 0$, the *exact* constraint:

$$ax \geq b,$$

where $a \in [\hat{a} - \overline{E_a}(\hat{a}), \hat{a} + \overline{E_a}(\hat{a})]$ and $b \in [\hat{b} - \overline{E_a}(\hat{b}), \hat{b} + \overline{E_a}(\hat{b})]$, is no longer represented by the *inexact* constraint:

$$\hat{a}x \geq \hat{b},$$

but is relaxed into the following corrected constraint:

$$(\hat{a} + \overline{E_a}(\hat{a}))x \geq \hat{b} - \overline{E_a}(\hat{b}). \tag{39}$$

While inequality (39) is valid only for $x \geq 0$, we can similarly derive a corrected constraint for any a, b, and x.

This data correction is carried out for all inequality constraints; equality constraints being considered as two opposite inequalities. The exact feasible region is extended because we are bound to consider the worst-case errors for all parameters: better consider a marginally extended feasibility region than an empty one.

Note that our primary goal is to deal with infeasibilities, that is why we focus on constraints. As the feasibility region is enlarged and the objective coefficients are not adjusted accordingly, it is possible that a greater objective value is found with data correction.

Note also that we propose to use scalings to deal with rounding errors (as described in Section 4.2.1) and data correction to handle data errors. It seems

like we could have used scalings for both. However, using scalings would not work with an exact solver because no feasibility tolerance is used. Moreover, feasibility tolerance can handle only cumulative errors on all terms of a constraint, while the proposed data correction handles each term individually.

We consider the simple model presented in Section 4.1. We choose to keep scaling C for volumes for all reservoirs of all instances as it was shown to be a suitable scaling. We sort the instances based on the following case disjunction:

- case 1 if QSoptEx and Cplex reach a feasible solution without data correction;
- case 2 if QSoptEx and Cplex find no solution with data correction;
- case 3 if QSoptEx and Cplex find no solution without data correction, but both solvers reach a feasible solution with it;
- case 4 if QSoptEx finds no solution without data correction, but Cplex reaches a feasible solution without it, and both solvers reach a feasible solution with data correction.

Table 5 summarizes feasibility consistency of the solutions obtained by Cplex and QSoptEx without data correction ($noDC$) and with data correction (wDC) for the 66 instances presented in Section 3.1.

Firstly, comparing QSoptEx results without or with data correction (second and fourth columns) shows that 11 (= 9+2) instances were infeasible because of data errors and become feasible with data correction. Secondly, QSoptEx and Cplex solutions disagree without data correction for 2 instances (case 4), whereas they now agree with it. All in all, the proposed data correction is shown to be effective as feasibility consistency between solvers is recovered and feasibility is also recovered for 11 more instances.

Let us compare relative objective error between solvers for the instances that are rightfully found feasible and solved to optimality: the 44 feasible instances without data correction, relative to case 1, and the 55 feasible (= 44 + 9 + 2) instances with data correction, relative to cases 1, 3, and 4. Table 6 summarizes

statistics on relative objective error according to data correction. The relative objective error $\%\Delta$ is computed as explained previously in Section 4.2.2. For the two sets of instances that are solved to optimality, we restate the number of instances and we present the same statistics as in Table 4.

Luckily, rounding errors are mitigated with data correction as CPLEX objective value is relatively more accurate. Although rounding errors and data errors — and the proposed repairs — may override one another depending on the problems, it remains necessary to implement the two reparations separately to deal with each kind of errors.

4.4. Extensions of results to the complete model

Ideally, we would like to check that the computational results obtained by scaling and data correction for the simple linear model can be extended to the complete mixed-integer model. We tried to use SCIP-EX [25], an MILP solver based on rational arithmetic. SCIP-EX uses SCIP 3.0.0's branch-and-bound framework and invokes QSOPTEX for LP solution. Unfortunately, SCIP-EX solution times are too long for the complete model for all instances but those relative to valley `v-a`. Interestingly SCIP-EX and CPLEX solutions for the complete model with data correction and scaling C are consistent in terms of feasibility for each instance relative to valley `v-a`. This seems to validate our approach eventhough we would need to further check on the whole set of instances.

Therefore, we do not refer to results of an exact solver to check feasibility consistency. We keep on using scaling C for volumes and data correction on the complete model. Table 7 summarizes the solution obtained by CPLEX for the 66 instances presented in Section 3.1 with the complete model described in Section 2 without and with data correction. The first (second, respectively) column shows solution status *noDC* (*wDC*, respectively). The third column gives the number of instances #inst for each combination of the first and second column entries. Note that results *noDC* correspond to the ones presented in Section 3.2 (Table 2).

Data correction helps to increase the number of feasible instances from 14 to 46 and allows to doubt the feasibility of 4 instances. In the sequel we assume the impact of numerical error is sufficiently reduced thanks to data correction and scaling C and that their impact can be neglected.

5. Dealing with model infeasibilities

Without numerical errors, we expect our problem to be modeled so as to be feasible. As it is not the case, we would like to understand the sources of infeasibilities and update our model accordingly. In this section, infeasibilities are first classified according to the two main specifications: discrete operations and target volume constraints (or water-management policies).

5.1. Classification of infeasibilities

Cplex's *conflict refiner* routine [21] offers to identify a "set of mutually contradictory constraints". It is not guaranteed to be minimal, that is to say it may not be an Irreducible Infeasible Subset (as defined in [26]). Even so we cannot systematically call the routine because it is very time consuming given the size of our instances. In addition, this routine considers all constraints individually while we are interested on insights by groups of constraints. Therefore, we sort of mimic the conflict refiner routine by manually relaxing sets of constraints corresponding to the two specifications of the complete model — namely discrete operations and target volume constraints — and by checking feasibility status. We still use it occasionally when individual refinement is needed.

Indeed, the specification of discrete operations is relaxed so as to obtain the simple model s from the complete model c. We also consider relaxations obtained by removing target volume bounds which are a hard expression of water-management policies. Ideally we wish to satisfy target volumes but we relax them to detect if they are sources of infeasibility and therefore obtain models $\tilde{s}$ and $\tilde{c}$. Target volume bounds will be referred to as TV.

According to the combination of relaxations, we consider four models:

- the simple model (from Section 4.1) without target volumes $\tilde{s}$
- the simple model s: $\mathcal{F}(s) \subset \mathcal{F}(\tilde{s})$
- the complete model (from Section 2.2) without target volumes $\tilde{c}$: $\mathcal{F}(\tilde{c}) \subset \mathcal{F}(\tilde{s})$
- the complete model c: $\mathcal{F}(c) \subset (\mathcal{F}(\tilde{c}) \cap \mathcal{F}(s))$ as $\mathcal{F}(\tilde{c})$ and $\mathcal{F}(s)$ are relaxation of $\mathcal{F}(c)$ obtained by relaxing two different sets of constraints, i.e., the target volume constraints and the specification of discrete operations, respectively

where $\mathcal{F}(m)$ is the feasible set of model m with respect to q and v variables.

The four models above allow us to classify instances infeasibility. In particular, infeasibilities for the complete model, i.e., $\mathcal{F}(c) = \emptyset$, can be classified according to the feasibility status of the considered relaxations:

- *data inconsistent* when $\mathcal{F}(\tilde{s}) = \emptyset$: it may happen, for example, when the initial volume is out of bounds $\Psi_0 > \overline{\Psi}$, when bounds are reversed $\underline{\Psi} > \overline{\Psi}$, when a unit is declared unavailable $\overline{Q} = \overline{\Sigma} = 0$ while its upstream reservoir overflows. There exist several other cases and the interpretation of an operator is often required to recover a relevant feasibility. As we want to propose a systematic treatment of infeasibilities, we decide to discard data inconsistencies from the scope of this work.
- *unattainable TV* when $\mathcal{F}(s) = \emptyset$ and $\mathcal{F}(\tilde{c}) \neq \emptyset$: it happens when the target volumes are unattainable even when discrete unit operational domains are relaxed. This kind of infeasibilities will be dealt with in Section 5.2.
- *impossible discrete operations* when $\mathcal{F}(\tilde{c}) = \emptyset$ and $\mathcal{F}(s) \neq \emptyset$: such infeasibilities happen when the sets of discrete operational points together with ramp-up/down, monotonicity and disjunctive constraints (22)-(23) forbid values of water flow required to remain within the reservoir bounds. For the same reasons as data inconsistencies, we discard this kind of infeasibilities.

- *unattainable TV & impossible discrete operations* when $\mathcal{F}(s) = \emptyset$ and $\mathcal{F}(\tilde{c}) = \emptyset$: when the two previous kinds of infeasibilities concomitantly happen. As we discard *impossible discrete operations*, we also discard this kind of infeasibilities.

- *incompatible TV & discrete operations* when $\mathcal{F}(s) \neq \emptyset, \mathcal{F}(\tilde{c}) \neq \emptyset$: it happens when the target volumes are incompatible with complete discrete unit operational domains, though target volumes are attainable with relaxed unit operational domains, and discrete operations are feasible without target volumes. Like the cases where target volumes are unattainable, this kind of infeasibilities will be dealt with in Section 5.2.

In Table 8, we refer to results obtained considering the full test set with the complete model, scaling C, and data correction. Namely, the first column shows CPLEX solution status, the second column shows (when applicable) infeasibility classification (as defined in details below), and the third column gives, for each combination of status/class, the number of relative instances #inst.

Within the tested set, there was no occurrence of *impossible discrete operations* and *unattainable TV & impossible discrete operations*. We are left with cases relative to *unattainable TV* and *incompatible TV & discrete operations*. We must therefore deal with too tight target volumes, which are responsible for the infeasibilities of at least 8 (=6+2) instances of our tests.

For the 7 instances found intractable for the complete model, solving the relaxed models reached feasible solutions: $\mathcal{F}(s) \neq \emptyset, \mathcal{F}(\tilde{c}) \neq \emptyset$. They can either be feasible or infeasible as incompatible between target volumes and discrete operations.

5.2. Reformulation to deal with unattainable target volumes

We choose to relax target volumes instead of tampering with discrete operations. Firstly, target volumes are introduced to control water level daily trajectories. They are needed either to enforce mid-term management policies when those cannot be properly reflected through water values, or to satisfy

punctual requirements for reservoir operations. In principle introducing target volumes should be regarded as complementary/strategic constraints rather than mandatory. Considering hard constraints as in (11)-(12) often results in infeasibility as seen in Section 5.1 and also as mentioned in, for example, [27]. Secondly, the discrete operations may be very critical, if they model prohibited operating zones as in [28] for example.

To amend target volumes, we propose a 2-stage method. In the first stage we complete and marginally modify the model formulated in Section 2 to recover feasibility. Slack variables are introduced, namely:

$\overline{\alpha}_r, \underline{\alpha}_r$ deviations to mid-horizon maximum and minimum target volumes of reservoir r ($\forall r \in R^T$) [m^3].

$\overline{\beta}_r, \underline{\beta}_r$ deviations to final maximum and minimum target volumes of reservoir r ($\forall r \in R^T$) [m^3].

The bounds on these new variables are defined as follows ($\forall r \in R^T$):

$$0 \leq \overline{\alpha}_r \leq \max(\overline{A}_r - \overline{\Psi}_r, 0) \tag{40}$$

$$0 \leq \underline{\alpha}_r \leq \max(\underline{\Psi}_r - \underline{A}_r, 0) \tag{41}$$

$$0 \leq \overline{\beta}_r \leq \max(\overline{B}_r - \overline{\Psi}_r, 0) \tag{42}$$

$$0 \leq \underline{\beta}_r \leq \max(\underline{\Psi}_r - \underline{B}_r, 0). \tag{43}$$

Relaxing constraints (11)-(12) as follows is expected to guarantee feasibility:

$$\underline{A}_r - \underline{\alpha}_r \leq v_{\overline{t}/2,r} \leq \overline{A}_r + \overline{\alpha}_r \qquad \forall r \in R^T \tag{44}$$

$$\underline{B}_r - \underline{\beta}_r \leq v_{\overline{t},r} \leq \overline{B}_r + \overline{\beta}_r \qquad \forall r \in R^T. \tag{45}$$

Yet we want to relax the constraints only when necessary and we do not want to leverage relaxations to gross larger revenues. For instance, if target volume bounds are attainable, no slack should be introduced. To deviate as little as possible from the initial target volume bounds, we compute the minimal slacks necessary and sufficient to recover feasibility. To do this, objective function

(29) is replaced by another objective, i.e., the minimization of the sum of (non-negative) deviations:

$$\varepsilon = \min \sum_{r \in R^T} (\overline{\alpha}_r + \overline{\beta}_r + \underline{\alpha}_r + \underline{\beta}_r) \tag{46}$$

and constraints (40)-(43) are added while constraints (11)-(12) are replaced by (44)-(45). The first stage terminates with a value $\varepsilon \geq 0$. Such a value can be considered as the 1-norm of the deviation vector $(\overline{\alpha}_r - \underline{\alpha}_r, \overline{\beta}_r - \underline{\beta}_r)_{r \in R^T}$, thus inducing a non-negative value for ε.

In the second stage, we still consider the slack variables along with their related constraints (44)-(45), the only constraints we consider relaxable, and we limit the sum of deviations as follows:

$$\sum_{r \in R^T} (\overline{\alpha}_r + \overline{\beta}_r + \underline{\alpha}_r + \underline{\beta}_r) \leq \varepsilon \tag{47}$$

where ε results from the first stage. Either the initial feasible set, *i.e.*, without deviations, was empty and it is enlarged within the computed deviation to contain at least one feasible solution after the first stage, or the initial feasible set was already non-empty and it remains unchanged when the minimum aggregated deviation ε is zero. In the second stage, we can then optimize the original objective function (29) over a feasible set now guaranteed to be non-empty. Note that this kind of relaxation (47) differs from what standards MILP solver do (39). In the first case, the new right-hand-side ε is computed by optimizing (46) subject to (40)-(45) and can take a positive, non negligible value. In the second case, the coefficients and right-hand-side modification is bounded by the relative data error (typically negligible) computed beforehand.

The proposed method could be interpreted as a lexicographic method to solve a bi-objective MILP where the first objective is minimizing the deviation with respect to the target volumes (46) and the second objective is the original objective function (29). For details on lexicographic methods for multi-objective optimization problems the reader is referred to [29], for example.

5.3. Computational results

Table 9 expands Table 8 to include results of the feasibility recovery stage with statistics per valley.
Table entries are:

v valley label

time average wall-clock solution time and standard deviation in seconds.

#nodes average number of branch-and-bound nodes and standard deviation in thousands

#inst feasible number of feasible instances, now including instances for which the stage-1 (46) does not converge to 0 ($\varepsilon > 0$)

gap (%) average relative gap and standard deviation (only for feasible instances)

#inst- $\varepsilon > 0$ number of instances for which the sum of deviations ε is stricly positive.

$\|\varepsilon\|$ **(%)** average normalized sum of deviations and standard deviation (only for instances for which $\varepsilon > 0$)

Note that to allow for comparison with results relative to first computations in Section 3, the first five entries in Table 2 and 9 match.

The 46 instances that were originally feasible remain unchanged as the computed deviation is zero. The feasibility recovery stage converges to 0 for one instance originally intractable, thus proving original feasibility. Indeed, the original problem and the stage-1 problem are somewhat different and MIP solution processes are sensitive to such differences. As for the other 6 originally intractable instances, a non-optimal integer solution was found within time limit in the feasibility recovery stage. Note that we have neither proven original feasibility because the corresponding objective value is strictly greater than zero ($\varepsilon > 0$) nor original infeasibility because the continuous lower bound is still zero

at termination. The 5 instances that are data inconsistent remain infeasible as they are left aside from the feasibility recovery stage. For the 8 (=6+2) instances that were originally infeasible because of target volumes, a non-zero deviation is computed to recover feasibility. All in all, feasibility is recovered for 61 instances out of 66.

For those 61 instances, Table 10 summarizes solution statistics of the second stage.

Table entries are:

v valley label

#inst feasible number of feasible instances as found from the the feasibility recovery stage

time average wall-clock solution time and standard deviation in seconds.

#nodes average number of branch-and-bound nodes and standard deviation in thousands

gap (%) average relative gap and standard deviation

$\|\boldsymbol{\varrho}\|$ **(%)** average normalized revenues and standard deviation (as derived in (30)).

Regarding infeasibilities and the necessity to compute target volume deviations, valleys `v-a` (surprisingly), `v-c`, and `v-e` seem problematic and no trend related to the size/topography of the valleys appears. We observe the (expected) trend that larger valleys gross more revenues, although several instances are not solved to optimality and small valleys could feature high-capacity units. As for solution performance, no pattern emerges either: the "hardest" valley `v-f` is solved well with respect to valleys `v-c` and `v-e`. Indeed, the average gap for `v-c` is large because, for several instances, stage-2 fails to converge to an integer feasible solution so we have to settle with the solution of stage-1 as back up. For `v-e`, although quite good solutions are found, it seems it takes time to close the gap as the time limit of 1200s is always reached.

6. Conclusions

In this work we considered feasibility issues of the hydro-unit commitment (HUC) relative to units along a valley. The problem was formulated as a mixed-integer linear program. Given real-world instances, we reformulated the model to make the problem feasible. Compared with a standard HUC problem, we considered two additional specifications, namely the power-flow curves feature discrete operational points and each reservoir level should meet mid-horizon and final target volumes. In practice, solving this HUC problem often encountered several infeasibility situations. We followed a step-by-step approach to detect and fix one source of infeasibility at a time, namely numerical errors and model infeasibilities. Numerical errors were analyzed by resorting to an exact solver. We derived a preliminary processing of data by scaling variables and marginally correcting constraints to eliminate the detrimental effects of numerical errors. We showed that the remaining infeasibilities originated in conflicting specifications between target volumes and discrete operational points. We proposed a preliminary processing of the model to eliminate such infeasibilities by implementing a 2-stage method. In the first stage, a minimal deviation from target volumes is estimated to make the problem feasible. In the second stage, the original HUC problem is solved with a possible deviation from the target volumes as derived in the first stage.

On an original test set of 66 real-world instances with only 14 feasible instances (21%), the proposed method attained feasibility for 61 instances (92%). More specifically, the preliminary processing of data and model helped recover/decide on the feasibility of respectively 32 and 15 instances out of the 52 problematic ones.

An in-depth analysis of the detrimental effects of rounding errors on solutions would be an interesting perspective to improve the condition number of the constraints matrix, thus extending the proposed preliminary processing of data.

For future work, we aim at studying new and efficient methods to find a good feasible solution of the HUC problem. Moreover, the more accurate knowledge

of the challenging aspects of the HUC problem allow us to tailor effective exact methods.

Acknowledgement

This work was supported by ANRT CIFRE number of contract 1106/2012. The third author acknowledge EDF for financial support to pursue this research. The authors are grateful to Grace Doukopoulos for initiating this research project and Arnaud Lenoir for his support with the numerical tests. We thank two anonymous referees for their useful comments and suggestions that helped to significantly improve the paper.

References

[1] A. Renaud, Daily generation management at Electricite de France: from planning towards real time, Automatic Control, IEEE Transactions on 38 (7) (1993) 1080–1093.

[2] M. Held, P. Wolfe, H. P. Crowder, Validation of subgradient optimization, Mathematical Programming 6 (1) (1974) 62–88. doi:10.1007/BF01580223.

[3] P. Wolfe, A method of conjugate subgradients for minimizing nondifferentiable functions, Springer Berlin Heidelberg, Berlin, Heidelberg, 1975, pp. 145–173.

[4] G. Hechme-Doukopoulos, S. Brignol-Charousset, J. Malick, C. Lemaréchal, The short-term electricity production management problem at EDF, Optima Newsletter-Mathematical Optimization Society 84 (2010) 2–6.

[5] G. Cohen, Auxiliary problem principle and decomposition of optimization problems, Journal of Optimization Theory and Applications 32 (3) (1980) 277–305.

[6] W. W.-G. Yeh, Reservoir management and operations models: A state-of-the-art review, Water Resources Research 21 (12) (1985) 1797–1818.

[7] J. W. Labadie, Optimal operation of multireservoir systems: State-of-the-art review, Journal of Water Resources Planning and Management 130 (2) (2004) 93–111. arXiv:http://dx.doi.org/10.1061/(ASCE)0733-9496(2004)130:2(93).

[8] R. Taktak, C. D'Ambrosio, An overview on mathematical programming approaches for the deterministic unit commitment problem in hydro valleys, Energy Systems (2016) 1–23.

[9] G. B. Sheble, G. N. Fahd, Unit commitment literature synopsis, Power Systems, IEEE Transactions on 9 (1) (1994) 128–135.

[10] N. P. Padhy, Unit commitment-a bibliographical survey, Power Systems, IEEE Transactions on 19 (2) (2004) 1196–1205.

[11] M. Tahanan, W. van Ackooij, A. Frangioni, F. Lacalandra, Large-scale unit commitment under uncertainty, 4OR 13 (2) (2015) 115–171.

[12] A. Arce, T. Ohishi, S. Soares, Optimal dispatch of generating units of the itaipu hydroelectric plant, Power Systems, IEEE Transactions on 17 (1) (2002) 154–158.

[13] A. J. Conejo, J. M. Arroyo, J. Contreras, F. A. Villamor, Self-scheduling of a hydro producer in a pool-based electricity market, Power Systems, IEEE Transactions on 17 (4) (2002) 1265–1272.

[14] A. Borghetti, C. D'Ambrosio, A. Lodi, S. Martello, An MILP Approach for Short-Term Hydro Scheduling and Unit Commitment With Head-Dependent Reservoir, Power Systems, IEEE Transactions on 23 (3) (2008) 1115–1124.

[15] E. Parrilla, J. García-González, Improving the b&b search for large-scale hydrothermal weekly scheduling problems, International Journal of Electrical Power & Energy Systems 28 (5) (2006) 339–348.

[16] Q. Zhai, X. Guan, F. Gao, A necessary and sufficient condition for obtaining feasible solution to hydro power scheduling with multiple operating zones, in: Power Engineering Society General Meeting, 2007. IEEE, 2007, pp. 1–7.

[17] M. Padberg, Approximating separable nonlinear functions via mixed zero-one programs, Operations Research Letters 27 (1) (2000) 1–5.

[18] J. Lee, J. Leung, F. Margot, Min-up/min-down polytopes, Discrete Optimization 1 (1) (2004) 77–85.

[19] D. Rajan, S. Takriti, Minimum up/down polytopes of the unit commitment problem with start-up costs, Tech. Rep. RC23628, IBM Research Division (2005).

[20] W. van Ackooij, R. Henrion, A. Möller, R. Zorgati, Joint chance constrained programming for hydro reservoir management, Optimization and Engineering 15 (2) (2014) 509–531.

[21] IBM, ILOG CPLEX 12.4 User's Manual, IBM (2012).

[22] N. J. Higham, Accuracy and stability of numerical algorithms, Society for Industrial and Applied Mathematics, 2002.

[23] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, K. Wolter, MIPLIB 2010, Mathematical Programming Computation 3 (2) (2011) 103–163.

[24] D. G. Espinoza, On linear programming, integer programming and cutting planes, Ph.D. thesis, Georgia Institute of Technology (2006).

[25] W. Cook, T. Koch, D. E. Steffy, K. Wolter, An Exact Rational Mixed-Integer Programming Solver, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 104–116.

[26] O. Guieu, J. W. Chinneck, Analyzing infeasible mixed-integer and integer linear programs, INFORMS J. on Computing 11 (1) (1999) 63–77.

[27] M. R. Piekutowski, T. Litwinowicz, R. J. Frowd, Optimal short-term scheduling for a large-scale cascaded hydro system, in: Power Industry Computer Application Conference, 1993. Conference Proceedings, 1993, pp. 292–298.

[28] N. Sinha, R. Chakrabarti, P. K. Chattopadhyay, Fast evolutionary programming techniques for short-term hydrothermal scheduling, IEEE Transactions on Power Systems 18 (1) (2003) 214–220.

[29] M. J. Rentmeesters, W. K. Tsai, K.-J. Lin, A theory of lexicographic multi-criteria optimization, in: Engineering of Complex Computer Systems, 1996. Proceedings., Second IEEE International Conference on, 1996, pp. 76–79.

Table 1: Test set characteristics per valley

Valley	$\|R\|$	$\|I\|$	$\|I^R\|$	$\|I^C\|$	#bin vars		#vars	
`v-a`	1	1	0	0	136	(75)	524	(75)
`v-b`	2	3	1	0	953	(149)	2017	(149)
`v-c`	5	6	0	1	1707	(670)	3835	(670)
`v-d`	5	6	1	1	3296	(806)	6006	(1732)
`v-e`	5	9	1	0	3974	(582)	7066	(582)
`v-f`	10	16	2	4	4453	(406)	10343	(453)

Table 2: Solution statistics per valley relative to first computations for the complete model

v	time (s)		#nodes		#inst	gap (%)		$\|\varrho\|$(%)		#inst-no sol.	
	avg	std	avg	std	feasible	avg	std	avg	std	infeas.	intrac.
v-a	0	0	0	0	2	0.0	0.0	50.0	70.5	9	0
v-b	475	587	2,621	3,167	6	0.0	0.0	35.7	32.1	0	5
v-c	546	627	2,316	2,964	0	-	-	-	-	6	5
v-d	843	548	1,155	856	2	0.0	0.0	50.0	70.5	2	7
v-e	546	627	666	791	0	-	-	-	-	6	5
v-f	656	626	560	573	2	0.0	0.0	50.0	70.7	5	4

Table 3: Number of instances whose solutions are (in)consistent according to scaling, for the simple model

feasibility	QSoptEx	Cplex status					#inst
consistency	status	scaling A	scaling B	scaling C	scaling D	scaling E	
consistentF	optimal	optimal	optimal	optimal	optimal	optimal	12
consistentI	infeasible	infeasible	infeasible	infeasible	infeasible	infeasible	18
inconsistentF	optimal	infeasible	optimal	optimal	optimal	optimal	32
inconsistentI	infeasible	optimal	optimal	optimal	optimal	optimal	1
	infeasible	infeasible	optimal	optimal	optimal	optimal	1
	infeasible	infeasible	infeasible	infeasible	infeasible	optimal	2
Total Result							66

Table 4: Statistics of the relative objective error according to scaling, for the simple model

	%Δ			
	scaling B	scaling C	scaling D	scaling E
max	5.7E-03	8.1E-09	1.7E-08	1.6E-08
avg	2.1E-04	1.9E-10	4.4E-10	4.1E-10
std	1.0E-03	1.2E-09	2.5E-09	2.4E-09

Table 5: Number of instances whose solutions are (in)consistent according to data correction, for the simple model with scaling C

case#	*noDC*		*wDC*		#inst
	QSoptEx	Cplex	QSoptEx	Cplex	
case 1	optimal	optimal	optimal	optimal	44
case 2	infeasible	infeasible	infeasible	infeasible	11
case 3	infeasible	infeasible	optimal	optimal	9
case 4	infeasible	optimal	optimal	optimal	2
Total Result					66

Table 6: Statistics of relative objective error according to data correction, for the simple model with scaling C

	%Δ	
	noDC	*wDC*
max	8.1E-09	9.7E-11
avg	1.9E-10	4.9E-12
std	1.2E-09	1.8E-11
#inst	44	55

Table 7: Number of instances according to Cplex solution status, without data correction *noDC* and with it *wDC*, for the complete model with scaling C

noDC	*wDC*	#inst
feasible	feasible	14
infeasible		11
intractable		21
infeasible	infeasible	13
infeasible	intractable	4
intractable		3
Total Result		66

Table 8: Number of instances according to solution status and infeasibility classification, for the complete model with scaling C and data correction

solution status	infeasibility classification	#inst
feasible	-	46
intractable	-	7
infeasible	data inconsistent	5
	unattainable TV	6
	impossible discrete operations	0
	unattainable TV & impossible discrete operations	0
	incompatible TV & discrete operations	2
Total Result		66

Table 9: Solution statistics per valley of first stage for the complete model with scaling C and data correction

v	time (s)		#nodes		#inst	gap (%)		#inst	$\|\varepsilon\|$(%)	
	avg	std	avg	std	feasible	avg	std	$\varepsilon > 0$	avg	std
v-a	0	0	0	0	7	0.0	0.0	1	10.9	0.0
v-b	0	0	0	0	11	0.0	0.0	0	0.0	0.0
v-c	661	620	2,430	3,081	11	50.3	48.2	7	5.3	5.3
v-d	112	361	100	330	10	11.1	33.3	1	0.4	0.0
v-e	253	115	188	106	11	16.8	11.3	5	27.5	41.7
v-f	7	5	1	2	11	0.0	0.0	0	0.0	0.0

Table 10: Solution statistics per valley of second stage for the complete model with scaling C and data correction

v	#inst	time (s)		#nodes		gap (%)		$\|\varrho\|$(%)	
	feasible	avg	std	avg	std	avg	std	avg	std
v-a	7	0	0	0	0.0	0.0	0.0	25.0	1.0
v-b	11	412	512	2,359	2,953	0.0	0.0	30.0	5.0
v-c	11	771	543	2,216	1,725	20.0	32.0	30.0	15.0
v-d	10	250	501	565	1,334	0.0	0.0	53.0	20.0
v-e	11	1,201	0	1,054	239	1.3	2.4	37.0	16.0
v-f	11	630	558	464	470	0.1	0.1	56.0	22.0