Tight and Compact MILP Formulation for the Thermal Unit Commitment Problem

Published in:
IEEE Transactions on Power Systems
Special Section on Analysis and Simulation of
Very Large Power Systems
Available: http://dx.doi.org/10.1109/TPWRS.2013.2251373

Germán Morales-España*, Jesus M. Latorre†, and Andres Ramos‡

Institute for Research in Technology (IIT)
Universidad Pontificia Comillas
Madrid, España

April 2013

*german.morales@iit.upcomillas.es; gmorales@kth.se
†jesus.latorre@iit.upcomillas.es
‡andres.ramos@iit.icai.upcomillas.es

Tight and Compact MILP Formulation for the Thermal Unit Commitment Problem

Germán Morales-España, *Student Member, IEEE,* Jesus M. Latorre, *Member, IEEE,* and Andres Ramos

***Abstract*—This paper presents a Mixed-Integer Linear Programming (MILP) reformulation of the thermal Unit Commitment (UC) problem. The proposed formulation is simultaneously tight and compact. The tighter characteristic reduces the search space and the more compact characteristic increases the searching speed with which solvers explore that reduced space. Therefore, as a natural consequence, the proposed formulation significantly reduces the computational burden in comparison with analogous MILP-based UC formulations. We provide computational results comparing the proposed formulation with two others which have been recognized as computationally efficient in the literature. The experiments were carried out on 40 different power system mixes and sizes, running from 28 to 1870 generating units.**

***Index Terms*—Mixed-integer linear programming, strong lower bounds, thermal units, unit commitment.**

NOMENCLATURE

Upper-case letters are used for denoting parameters and sets. Lower-case letters denote variables and indexes.

A. Indexes and Sets

$g \in \mathcal{G}$	Generating units, running from 1 to G.
$s \in \mathcal{S}_g$	Startup segments, running from 1 (hottest) to S_g (coldest), see Fig. 1.
$t \in \mathcal{T}$	Hourly periods, running from 1 to T hours.

B. Parameters

C_g^{LV}	Linear variable cost of unit g [$/MWh].
C_g^{NL}	No-load cost of unit g [$/h].
C^{NSE}	Non-served energy cost [$/MWh].
C_g^{SD}	Shutdown cost of unit g [$].
C_{gs}^{SU}	Coefficients of the startup cost function of unit g, see Fig. 1 [$].
D_t	Load demand in hour t [MW].
$\overline{P}_g$	Maximum power output of unit g [MW].
$\underline{P}_g$	Minimum power output of unit g [MW].
R_t	Spinning reserve requirement in hour t [MW].
RD_g	Ramp-down rate of unit g [MW/h].
RU_g	Ramp-up rate of unit g [MW/h].
SD_g	Shutdown capability of unit g [MW].
SU_g	Startup capability of unit g [MW].
TD_g	Minimum downtime of unit g [h].
TU_g	Minimum uptime of unit g [h].
T_{gs}^{SU}	Times defining the segment s limits, $\left[T_{gs}^{SU}, T_{g,s+1}^{SU}\right)$, of the startup cost function of unit g [h], see Fig. 1.

The authors are with the Institute for Research in Technology (IIT) of the School of Engineering (ICAI), Universidad Pontificia Comillas, Madrid, España (e-mail: german.morales@iit.upcomillas.es; gmorales@kth.se; jesus.latorre@iit.upcomillas.es; andres.ramos@upcomillas.es).

C. Variables

1) Positive and Continuous Variables:

nse_t	Non-served energy in hour t [MWh].
p_{gt}	Power output at hour t of unit g, production above the minimum output $\underline{P}_g$ [MW].
r_{gt}	Spinning reserve provided by unit g in hour t [MW].

2) Binary Variables:

u_{gt}	Commitment status of the unit g for hour t, which is equal to 1 if the unit is online and 0 offline.
v_{gt}	Startup status of unit g, which takes the value of 1 if the unit starts up in hour t and 0 otherwise.
w_{gt}	Shutdown status of unit g, which takes the value of 1 if the unit shuts down in hour t and 0 otherwise.
δ_{gst}	Startup-type s of unit g, which takes the value of 1 in the hour where the unit starts up and has been previously offline within $\left[T_{gs}^{SU}, T_{g,s+1}^{SU}\right)$ hours, see Fig. 1.

I. INTRODUCTION

A. Motivation

EFFICIENT resource scheduling is necessary in power systems to achieve an economical and reliable energy production and system operation, either under centralized or competitive environments. This can be achieved by solving the Unit Commitment (UC) problem, of which the main objective is to minimize the total system operational costs while operating the system and units within secure technical limits [1]–[3].

Mixed-Integer Linear Programming (MILP) has become a very popular approach to solving UC problems due to significant improvements in off-the-shelf MILP solvers, based on the branch-and-cut algorithm. The combination of pure algorithmic speedup and the progress in computer machinery has meant that solving MILPs has become 100 million times faster over the last 20 years [4]. Recently, the world's largest competitive wholesale market, PJM, changed from Lagrangian Relaxation to MILP to tackle its UC-based scheduling problems [5]. There is an extensive literature comparing the pros and cons of MILP with its competitors [2], [6].

Despite the significant improvements in MILP solving, the time required to solve UC problems continues to be a critical limitation that restricts the size and scope of UC models. Nevertheless, improving an MILP formulation can dramatically reduce its computational burden and so allow the implementation of more advanced and computationally demanding problems, such as stochastic formulations [7],

accurate modelling of different types of (online and offline) reserves [8], or transmission switching [9].

B. Literature Review

1) Performance of MILP Formulations —Tightness vs. Compactness: The branch-and-cut algorithm solves MILP by solving a sequence of Linear Program (LP) relaxations. The LP relaxation of a MILP problem is obtained by relaxing its integrality requirements. During the solving process (branching), upper bounds (feasible integer solutions) and lower bounds (LP relaxations) are computed. The quality of a feasible integer solution is measured with the optimality tolerance, which is the difference between upper and lower bounds. In order to reduce this difference, upper bounds are decreased by finding better integer solutions (e.g. heuristics) and lower bounds are increased by strengthening the LP relaxation (e.g. adding cutting planes) [10]. Providing an MILP formulation with strong lower bounds (LP relaxation near to the optimal integer solution) can dramatically reduce the length of the search for optimality [11], [12]. In addition, strong lower bounds effectively guide the search for better upper bounds (i.e. heuristics explore the neighbourhood of the LP relaxation to find potentially better integer solutions).

The computational performance of an MILP formulation is mainly influenced by its tightness (distance between relaxed and integer solutions) and compactness (quantity of data to process when solving the problem). These two characteristics are actually fully exploited by off-the-shelf MILP solvers. Even though solvers' breakthrough is due to the synergy between different strategies (e.g. heuristics, cuts, node presolve), introducing cutting planes has been recognized as the most effective strategy, followed by root presolve [13], [14]. The former strategy dynamically tightens the formulation around the integer feasible solution point. The latter makes the initial problem formulation more compact (by removing redundant variables and constraints) and also tighter (by strengthening constraints and variable bounds).

The tightness of an MILP formulation defines the search space (relaxed feasible region) that the solver needs to explore in order to find the (optimal integer) solution. A given MILP problem has many possible formulations. If F1 and F2 are two formulations for the same MILP problem, and the feasible region of F1 is contained inside the feasible region of F2, then F1 is a tighter formulation than F2, and thus the lower bound provided by the LP relaxation of F1 is always greater than or equal to that provided by F2 [11], [15]. That is, F1 provides stronger lower bounds and the optimal solution of its LP relaxation is nearer to the optimal integer solution.

The compactness of an MILP formulation refers to its size and defines the searching speed that the solver takes to find the optimal solution, since during the process, many LP relaxations are repeatedly solved. Although, the number of constraints is considered to be the best simple predictor of the LP models' difficulty [16], [17], the number of nonzeros also has a significant impact on solution times [10]. Therefore, formulation F1 is considered more compact than F2 if F1 presents simultaneously fewer constraints and nonzeros than F2.

Research on improving MILP formulations is usually focused on tightening rather than on compacting. An MILP formulation is typically tightened by adding a huge number of constraints, which increases the problem size [18], [19]. Although this tightening reduces the search space, solvers may take more time exploring it because they are now required to repeatedly solve larger LPs. Consequently, when a formulation is tightened while significantly affecting its compactness, a more compact and less tight formulation may be solved faster, because the solver is able to explore the larger feasible region more rapidly [18]. On the other hand, compact formulations usually provide weak (not strong) lower bounds. In conclusion, creating tight or compact computationally efficient formulations is a non trivial task because the obvious formulations are very weak (not tight) or very large, and trying to improve the tightness (compactness) usually means harming the compactness (tightness).

2) Improving UC formulations: Improving MILP formulations, especially the tightness, has been widely researched. In fact, all the cutting plane theory, which has meant the breakthrough in MILP solving, is about tightening the formulations [4], [10], [14], [20]. In the case of UC problems, there have been efforts affecting specific aspects. Ref. [21] reduced the number of binary variables, claiming that this speeded up the search process compared with the 3-binary models [2], [6]. In [22], a strong formulation of the minimum up/down time constraints is proposed; in [23], a tighter linear approximation for quadratic generation costs is described; Ref. [19] presents a new class of inequalities giving a tighter description of the feasible operating schedules for generators; and [24] proposes a tight and compact formulation for the startup and shutdown unit's power trajectories.

From the aforementioned formulations, [21], [22] and [19] have focused on improving the basic technical constraints (e.g., ramping limits, generation limits, minimum up/down times). As stated in [19], the main disadvantage of [21] is that avoiding the startup and shutdown variables hinders the possibility to generate and use strong valid inequalities, such as the minimum up/down time constraints proposed in [22]. Ref. [19] overcomes this problem by using the 3-binary format and thus introducing many additional inequalities (using all the binaries) to tighten the UC formulation. However, the main drawback of [19] is that it creates a huge model where, in order to obtain a computational advantage, the additional inequalities need to be appropriately introduced to the formulation during the solving process (dynamically). Ref. [19] also presents the additional disadvantage of implementation complexity, where the modeller needs to make an *ad-hoc* configuration of the solving strategy of MILP solvers to dynamically introduce these inequalities.

C. Contributions

This paper presents an alternative UC reformulation that describes the same basic UC problem as in [21] and [19]. In other words, we provide a formulation containing the same feasible integer solutions as those in [21] and [19], and hence obtaining the same optimal results.

The main contribution of this paper is three-fold:

1) A tight MILP formulation for the thermal UC problem is proposed in order to decrease the computational burden of analogous MILP formulations [19], [21].
2) The formulation is tightened at the same time as it is made more compact compared to both [21] and [19], hence overcoming the main disadvantage of usual tightening strategies [19]. The simultaneous tight and compact characteristics reinforce the convergence speed by reducing the search space and at the same time increasing the searching speed with which solvers explore that reduced space.
3) This reformulation can be used as the core of any UC problem, whether under centralized or competitive environments, from self-scheduling to centralized auction-based market clearing.

Furthermore, given the compactness of the formulation, additional extensions of the UC model will be less cumbersome. For example, in the case of a stochastic formulation the compactness and tightness can be fully exploited given the size and computational complexity of solving a large-scale stochastic UC problem. That is, a stochastic problem replicates the original deterministic structure to represent uncertainty; consequently, any reformulation will benefit from the characteristics of the deterministic formulation.

D. Paper Organization

The remainder of this paper is organized as follows: Section II details the UC reformulation. Section III provides and discusses results from several case studies, where a comparison with two other UC formulations is made. Finally, some relevant conclusions are drawn in Section IV.

II. Mathematical Formulation

This section details the reformulation of a typical UC. The constraints presented here characterize the same UC problem as those in [21] and [19] (see [25]). Hourly time intervals are considered, but it should be noted that the formulation can be easily adapted to handle shorter time periods.

A. Objective Function

The UC problem seeks to minimize the power system operation costs, which are defined as the sum of (i) the production cost, (ii) startup cost and (iii) shutdown cost. In addition, in order to take into account situations in which there is a lack of energy, (iv) the non-served energy cost is also included

$$\min \sum_{g\in\mathcal{G}}\sum_{t\in\mathcal{T}}\Bigg[\underbrace{C_g^{NL}u_{gt}+C_g^{LV}\left(\underline{P}_g u_{gt}+p_{gt}\right)}_{i} + \underbrace{\sum_{s\in\mathcal{S}_g} C_{gs}^{SU}\delta_{gst}}_{ii}+\underbrace{C_g^{SD}w_{gt}}_{iii}+\underbrace{C_t^{NSE}nse_t}_{iv}\Bigg]. \quad (1)$$

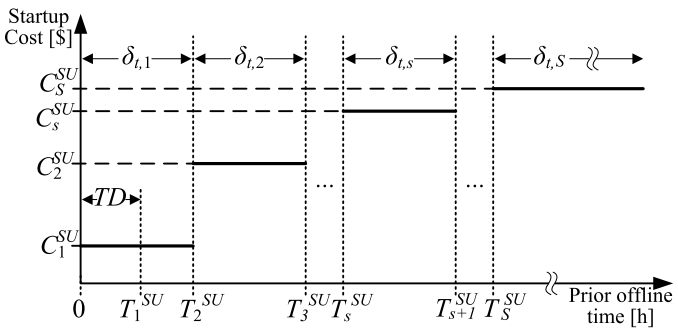


Fig. 1: Startup costs as a function of the unit's previous offline time.

1) Production Cost: The production cost is usually expressed as a quadratic function of the power output. Typically, this cost is modelled as a piecewise-linear function [21]. A tight formulation for this piecewise-linear approximation is given in [23]. This paper focuses on the reformulation of the unit's technical constraints as well as of the exponential startup cost. Therefore, for the sake of simplicity, we represent the production cost as a linear function. Note in (1) that the linear variable cost multiplies the total power output, which is the minimum output $\underline{P}_g u_{gt}$ plus the production above that minimum p_{gt}.

2) Startup Cost: Fig. 1 shows a typical exponential startup cost function [26], where C_{gs}^{SU} is the cost incurred when the unit g has been offline within the interval $\left[T_{gs}^{SU}, T_{g,s+1}^{SU}\right)$. This function is discrete since the time span has also been discretized into hourly periods. Refs. [21] and [19] represent this startup-cost function using the formulation proposed in [27]. We represent the same cost function using (2) and (3). As presented in our previous work [24], the startup-type variable δ_{gst} that activates the cost C_{gs}^{SU} in the objective function is selected by:

$$\delta_{gst} \le \sum_{i=T_{gs}^{SU}}^{T_{g,s+1}^{SU}-1} w_{g,t-i} \qquad \forall g, t\in\left[T_{g,s+1}^{SU}, T\right], s\in[1,S_g) \quad (2)$$

$$\sum_{s\in\mathcal{S}_g}\delta_{gst} = v_{gt} \qquad \forall g,t \quad (3)$$

where (2) stands for the time passed since the last shutdown and (3) ensures that only one startup cost value is selected when the unit actually starts up.

Note that (2) does not bound the coldest startup-type $\delta_{g,S_g,t}$. However, if the unit starts up at hour t and has been offline for at least $T_{S_g}^{SU}$ hours then constraints (2) and (3) ensure $\delta_{g,S_g,t}=1$. As discussed in [24], the variables δ_{gst} take binary values even if they are defined as continuous variables. This is due to the convex (monotonically increasing) characteristic of the exponential startup-cost function of thermal units [26] (see Fig. 1).

Due to the minimum downtime constraint (see Section II-B2), the hottest startup-type is only possible within the interval $\left[TD_g, T_{g,2}^{SU}\right)$. Therefore, constraint (2) is made more compact by defining $T_{g,1}^{SU}=TD_g$, see Fig. 1. Note that (2) is not defined for the first hours. Appendix A details how to

obtain δ_{gst} for the first hours depending on the unit's initial conditions.

An easy way to observe that this startup-cost formulation is tighter than the formulation in [19], [21] and [27] is that (2) and (3) provide upper bounds to the possible startup-cost values. For example, if the unit does not start up ($v_{gt}=0$) then (3) forces the startup cost to be zero. This is in contrast to [21], which does not provide upper bounds to the startup cost variable and then the objective function always has to look for the lowest feasible value. In addition, the computational results in Section III shows that the integrality gap of the UC formulation is significantly lowered when modelling the startup costs using (2) and (3).

B. Power-System and Thermal-Unit Constraints

1) Power System Requirements: The following constraints guarantee the balance between generation and load, and the provision of spinning reserve:

$$\sum_{g\in\mathcal{G}}\left[\underline{P}_g u_{gt}+p_{gt}\right]=D_t-nse_t \qquad \forall t \quad (4)$$

$$\sum_{g\in\mathcal{G}} r_{gt}\geq R_t \qquad \forall t \quad (5)$$

where the non-served energy variable nse_t is included to take into account situations of lack of energy. Note in (4) that power (MW) and energy (MWh) units are mixed, which is numerically correct in models with hourly time intervals. However, the time-period duration must be included when considering different time intervals.

2) Minimum Up and Downtime: The minimum number of periods that the unit must be online and offline are ensured with [22]:

$$\sum_{i=t-TU_g+1}^{t} v_{gi}\leq u_{gt} \qquad \forall g,t\in[TU_g,T] \quad (6)$$

$$\sum_{i=t-TD_g+1}^{t} w_{gi}\leq 1-u_{gt} \qquad \forall g,t\in[TD_g,T]\,. \quad (7)$$

This formulation provides strong lower bounds in comparison with others [9], [19], [22], as also shown in Section III. Appendix A describes how the initial conditions force the unit to remain online or offline during the first hours.

3) Logical Constraint: Equation (8) guarantees that v_{gt} and w_{gt} take the appropriate values when the unit starts up or shuts down.

$$u_{gt}-u_{g,t-1}=v_{gt}-w_{gt} \qquad \forall g,t. \quad (8)$$

The minimum up/down constraints ensure that a unit cannot start up and shut down simultaneously: note that (6) and (7) guarantee (dominate over) the inequalities $v_{gt}\leq u_{gt}$ and $u_{gt}\leq 1-w_{gt}$, respectively, which combined become $v_{gt}+w_{gt}\leq 1$. In addition, if u_{gt} is defined as a binary variable, (8) forces v_{gt} and w_{gt} to take binary values even if they are defined as continuous.

4) Generation Limits: The total unit production is modelled in two blocks: the minimum power output $\underline{P}_g$ that is generated just by being committed, and the generation over that minimum p_{gt}. The generation limits over the power output and the spinning reserve contribution are set as follows:

$$p_{gt}+r_{gt}\leq\left(\overline{P}_g-\underline{P}_g\right)u_{gt}-\left(\overline{P}_g-SU_g\right)v_{gt} \qquad \forall g\in\mathcal{G}^1,t \quad (9)$$

$$p_{gt}+r_{gt}\leq\left(\overline{P}_g-\underline{P}_g\right)u_{gt}-\left(\overline{P}_g-SD_g\right)w_{g,t+1} \qquad \forall g\in\mathcal{G}^1,t \quad (10)$$

where p_{gt} and r_{gt} are also constrained by the unit startup and shutdown capabilities.

Note that (9) and (10) are only applied for the subset $\mathcal{G}^1$, which is defined as the units in $\mathcal{G}$ with $TU_g=1$. For the cases in which $TU_g\geq 2$, both constraints can be replaced by a tighter and more compact formulation:

$$p_{gt}+r_{gt}\leq\left(\overline{P}_g-\underline{P}_g\right)u_{gt}-\left(\overline{P}_g-SU_g\right)v_{gt}-\left(\overline{P}_g-SD_g\right)w_{g,t+1} \qquad \forall g\notin\mathcal{G}^1,t. \quad (11)$$

Constraint (11) is tighter than (9) and (10) because in the event that the unit is online for just one period, the right side of (11) can be negative. Consequently, (11) is not valid for units with $TU_g=1$, because this constraint means that it is not feasible to operate the unit for just one online period. That is, (11) presents a tighter feasible region than (9) and (10) together. In addition, constraint (11) is also more compact because: 1) the two constraints (9) and (10) are lowered to one, and 2) constraint (11) introduces five non-zero elements (per unit and per period) in the constraint matrix in comparison with the eight elements introduced by (9) and (10) together.

Note that (9) and (10) are valid in both cases when $g\notin\mathcal{G}^1$ and when $g\in\mathcal{G}^1$. However, if (11) is used for $g\notin\mathcal{G}^1$, the same solution is obtained at the same time as the formulation is made more compact and tighter. Hence, the combination of both groups of constraints is used in this model.

One simple way to observe that the proposed formulation is tighter than those in [21] and [19] is by checking the upper bound of the constraints. For example, note that the upper bounds of (9)-(11) decrease when the binary variables v_{gt} and w_{gt} are different from zero. This is in contrast to [21] and [19], where the bounds of the constraints increase if $v_{gp},w_{gp}\neq 0$, see Appendix B for further details. In fact, the set of constraints (6)-(11) are the tightest possible representation (convex hull) for a unit operation without ramp constraints, although the mathematical proof for this is outside the scope of this paper.

5) Ramping Limits: The following constraints ensure that the unit operates within the ramp rate limits:

$$(p_{gt}+r_{gt})-p_{g,t-1}\leq RU_g \qquad \forall g,t \quad (12)$$

$$-p_{gt}+p_{g,t-1}\leq RD_g \qquad \forall g,t \quad (13)$$

where (12) guarantees that the unit can provide spinning reserve without violating the upwards ramp limit. The reader is referred to [8] for a more accurately modelling of different (online and offline) types of reserves.

III. Numerical Results

This section is divided into four parts. The first part describes the different UC formulations that were implemented. The different case studies are detailed in the second part. The third part presents a comparison between all the UC formulations, in terms of size and computational performance. Finally, the last part assess the impact on computational performance due to different number of binary variables in the formulations.

A. UC Formulations

To assess the computational burden of the proposed model, we compare it with those UC formulations in [21] and [19]. These two formulations have been recognized as computationally efficient in the literature [9], [19], [21], [22], [28]. For the sake of simplicity, the production costs are considered linear for all the formulations.

The following four formulations are then implemented:

1bin: This formulation is presented in [21] and requires a single set of binary variables (one per unit and per period), i.e., the startup and shutdown decisions are expressed as a function of the commitment decision variable.

3bin: The minimum up/down time constraints proposed in [22] (see Section II-B2) are implemented with the 3-binary equivalent formulation of [21]. This formulation is presented in [19] (see also [25]). Unlike [19], *3bin* is implemented without the extra cuts.

P1: This formulation is the same as *3bin*; however, the exponential startup-cost constraints presented in Section II-A2 were used instead of that in [27], which is the formulation used by [21] and [19].

P2: This is the complete formulation proposed in this paper.

It is important to note that all the formulations were implemented using the same objective function and the same set of constraints as the formulation presented in Section II (see also [25]). As a result, all of them describe the same optimization problem. The difference between them is how the constraints are formulated. In other words, for a given case study, all the formulations obtain the same optimal results, e.g., commitments, generating outputs and operation costs, as numerically shown in Section III-D.

Note that we define four sets of variables as binary u_{gt}, v_{gt}, w_{gt} and δ_{gst}. Thus, reducing the computational burden of the formulations, as discussed in Section III-D.

Although we focus the discussion on the performance of the proposed formulation *P2*, *P1* is implemented to observe the improvements in the proposed startup-cost formulation (see Section II-A2). Furthermore, when comparing *P2* with *P1*, one can clearly observe the additional improvements which result from considering the generation output variable above $\underline{P}_g$.

B. Case Studies

1) Set of Experiments: The following case-study based on the power system data in [21] is conducted to assess the computational performance of the proposed UC formulation.

Table I: Number of generators per problem case

Small cases										Large Cases									
	Generator								Total		Generator								Total
Case	1	2	3	4	5	6	7	8	Gens	Case	1	2	3	4	5	6	7	8	Gens
1	12	11	0	0	1	4	0	0	28	11	46	45	8	0	5	0	12	16	132
2	13	15	2	0	4	0	0	1	35	12	40	54	14	8	3	15	9	13	156
3	15	13	2	6	3	1	1	3	44	13	50	41	19	11	4	4	12	15	156
4	15	11	0	1	4	5	6	3	45	14	51	58	17	19	16	1	2	1	165
5	15	13	3	7	5	3	2	1	49	15	43	46	17	15	13	15	6	12	167
6	10	10	2	5	7	5	6	5	50	16	50	59	8	15	1	18	4	17	172
7	17	16	1	3	1	7	2	4	51	17	53	50	17	15	16	5	14	12	182
8	17	10	6	5	2	1	3	7	51	18	45	57	19	7	19	19	5	11	182
9	12	17	4	7	5	2	0	5	52	19	58	50	15	7	16	18	7	12	183
10	13	12	5	7	2	5	4	6	54	20	55	48	18	5	18	17	15	11	187

Table II: Sets of experiments

	Small cases		Large cases	
	Total gens	Time span	Total gens	Time span
$\times$7-*day*	28 to 54	7 days	132 to 187	7 days
$\times$10-*gen*	280 to 540	1 day	1320 to 1870	1 day

As presented in [19], an eight-generator data set is replicated to create larger instances and different power-systems mixes. These generation mixes, also used in [19], are shown in Table I. The replication introduces symmetry in the problems which makes them harder to solve than usual. The spinning reserve requirement of 10% of the power demand has to be met for each hour. The non-served energy cost is considered to be 1000 $/MWh. For quick reference, the generator data and power demand can be found in Appendix C.

Two test-sets are created in order to obtain even larger instances:

$\times$7-*day*: For the first test-set, the problem is solved with all the 20 instances presented in Table I for a time span of 7 days. The demand for the last two days (the weekend) is considered to be 80% of the demand on a working day.

$\times$10-*gen*: For the second test-set, the 20 different power-system mixes in Table I are replicated 10 times with a time span of one day. That is, the total number of generators for this test-set goes from 280 up to 1870.

Table II shows the different groups for all 40 cases that are considered here, where there are small and large cases for the two previous test-sets.

All tests were carried out using CPLEX 12.4 under GAMS [29] on a quad-core Intel-i7 2.4-GHz personal computer with 4 GB of RAM memory. The small and large cases, for both $\times$7-*day* and $\times$10-*gen*, are solved within 0.1% and 1.0% of relative optimality tolerance and a CPU time limit of one hour and 10 hours respectively. CPLEX defaults were used for all the experiments.

2) Performance Metrics: In order to summarize comparison results between formulations, geometric means of ratios are used since arithmetic means can be quite misleading when applied to a set of ratios [10], [14]. That is, a number of model characteristics between two formulations, e.g. number of constraints or runtimes, are compared using ratios, and the geometric mean over a set of case studies is then used as a performance metric. For example, when comparing solution

Table III: Problem Size (selected instances)

	Case	# of constraints ($\times 10^3$)				# of nonzero elements ($\times 10^6$)				# real var ($\times 10^3$)			# binary var ($\times 10^3$)		
		1bin	*3bin*	*P1*	*P2*	*1bin*	*3bin*	*P1*	*P2*	*1bin*	*3bin*	*P**	*1bin*	*3bin*	*P**
×7-day	1	104	104	51	37	0.74	0.75	0.25	0.22	19	14	10	5	14	24
	10	177	177	99	73	1.13	1.15	0.44	0.39	36	27	18	9	27	45
	11	454	453	241	178	3.09	3.13	1.11	0.98	89	67	45	22	67	111
	20	639	637	342	250	4.22	4.28	1.56	1.37	126	94	63	31	94	157
×10-gen	1	148	144	67	47	0.92	0.92	0.30	0.26	27	20	13	7	20	33
	10	252	247	132	95	1.44	1.45	0.56	0.49	52	39	26	13	39	64
	11	647	633	319	229	3.88	3.89	1.39	1.21	127	95	63	32	95	158
	20	910	891	453	323	5.35	5.36	1.96	1.71	180	135	90	45	135	222

* P1 is equal to P2 for these cases

Table IV: Problem size summary compared with *1bin* (%)

Case	# constraints			# nonzero elements			# real var		# binary var	
	3bin	*P1*	*P2*	*3bin*	*P1*	*P2*	*3bin*	*P**	*3bin*	*P**
Mean	98.8	51.0	36.9	100.8	36.1	31.7	75.0	50.1	300	497.3
min	98.6	47.1	33.7	100.4	32.8	28.8	75.0	50.0	300	495.9
max	98.9	54.8	39.9	101.1	39.8	34.9	75.1	50.2	300	498.9

* *P1* is equal to *P2* for these cases

times between *P1* and *1bin*, for each case study, two runtimes are obtained, one for *P1* and one for *1bin*. Given the set of runtimes, ratios are computed dividing the runtimes of *P1* by the corresponding runtime of *1bin*. Finally, the geometric mean is computed over these ratios and multiplied by 100 to obtain percentages. Thus, a geometric mean of ratios lower than 100% means that *P1* is faster. For the sake of brevity, the summary of results is presented in this paper; however, the different formulations and the set of statistics are presented in [25].

The formulation *1bin* is used as a benchmark to obtain the ratios. In other words, the formulations *P1*, *P2* and *3bin* (in the numerator) are compared with *1bin* (in the denominator) unless otherwise specified.

C. Comparing Different Formulations

1) Problem Size: Table III shows the dimensions for all the formulations for four selected instances. This sample is composed of the smallest and largest instances for the small cases (case 01 and 10) and large cases (case 11 and 20). There are instances which consist of almost a million constraints, present millions of nonzero elements in the constraint matrix, and contain 100000+ real and binary variables.

Table IV summarizes the different model sizes for all 40 instances in comparison with *1bin*. This summary is obtained as a geometric mean of ratios as described in Section III-B2. Note that *3bin* has almost as many constraints and non-zero elements as *1bin*. However, *3bin* presents three times more binary variables due to the explicit modelling of the startup and shutdown decisions as binary variables.

Although *P1* and *P2* present around five times as many binary variables as *1bin*, the number of constraints and nonzero elements was approximately reduced by two thirds. Consequently, *P2* is considerably more compact than *1bin* and *3bin* with respect to the nonzero elements and constraints (as mentioned in the Introduction).

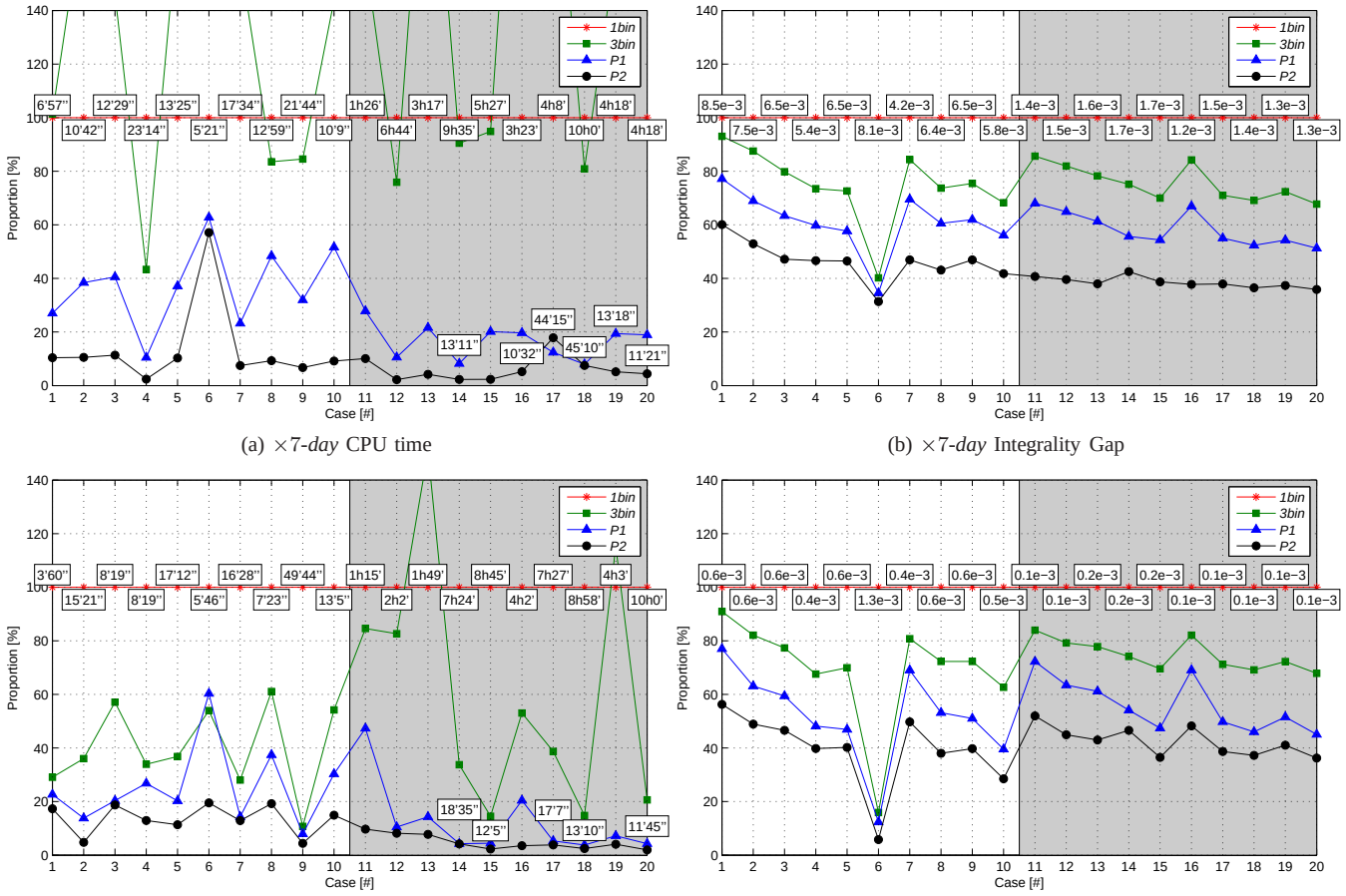
Note in Table IV that the main improvements in compactness are due to the proposed startup-cost formulation *P1*. Finally, *P2* further reduces the formulation size by modelling the power output variable above $\underline{P}_g$.

2) Computational Performance: Although the proposed formulation is more compact than *1bin* and *3bin*, this does not necessarily lead to a better computational performance. In fact, a compact formulation usually presents a weak (not tight) LP relaxation that can dramatically increase the MILP resolution time, as mentioned in the Introduction. The tightness of an MILP can be measured with the integrality gap [19]. The integrality gap is defined as $(Z_{MILP}-Z_{LP})/Z_{MILP}$, where Z_{LP} is the optimal value of the (initial) relaxed LP problem, and Z_{MILP} is the optimal integer solution. In practice, the problems are not solved until optimality but within an optimality tolerance. Therefore, for a given case study, Z_{MILP} is considered to be the best integer solution that was found among the four formulations after that case study was solved.

Fig. 2 shows the CPU times and integrality gaps for all the formulations and all the instances in comparison with *1bin* (using ratios), where *1bin* always represents 100%. The CPU times and the integrality gaps of *1bin* are shown within the squares to give an idea of the different problem magnitudes. The instances that took more than ten minutes to solve for *P2* can also be found within squares. The computational performance summary for the different formulations is presented in Table V for the small cases, large cases and all the cases together. Table V also shows the final optimality tolerance achieved as well as nodes explored and iterations.

In short, Fig. 2 and Table V shows that the proposed formulation *P2* considerably reduces the computational burden in comparison with *1bin* and *3bin* while achieving better solution qualities (lower final optimality tolerances) at the same time. Nevertheless, several aspects are worth mentioning:

1) *Relationship between integrality gap and runtime:* Figs. 2b and 2d show that *P2* always presents the lowest integrality gaps, followed by *P1*, *3bin* and finally *1bin*. Table V shows that the average time taken by the different formulations presents a similar pattern as to that of the integrality gaps, with the exception of *3bin* for the $\times$7-*day* test-set, which presented a worse performance than *1bin* (on average, *3bin* required more than 30% of the time that *1bin* required to solve all the $\times$7-*day* test-set). This is an unexpected result which can be explained as follows. In theory, lower integrality gaps lead to faster solving times when two formulations have similar sizes. However, in practice, MILP solvers use heuristics and cuts (among others) that may also dramatically influence performance. In addition, the enumeration tree and branching strategies change completely when formulations with different integer variables are compared. *1bin* presents fewer binary variables than *3bin*, and this is an advantage for finding integer feasible solutions. In general, a high number of integer variables complicates the search for feasible solutions. For the $\times$10-*gen* test-set, *3bin* showed a significant improvement over *1bin*: *3bin* required, on average, 40.6% of the CPU time needed by *1bin*. Note that in overall, *3bin* performs better for the small cases than for the large cases. A



(a) $\times$7-*day* CPU time

(b) $\times$7-*day* Integrality Gap

(c) $\times$10-*gen* CPU Time

(d) $\times$10-*gen* Integrality Gap

Fig. 2: Improvements in comparison with *1bin* (%). White areas correspond to small cases and gray to the large cases.

Table V: Computational Performance compared with *1bin* (%)

		CPU Time			Integrality Gap			Opt. Tolerance			Nodes			Iterations		
		3bin	*P1*	*P2*	*3bin*	*P1*	*P2*	*3bin*	*P1*	*P2*	*3bin*	*P1*	*P2*	*3bin*	*P1*	*P2*
$\times$7-*day*	Cases 01-10	120.9	33.7	9.7	73.3	59.8	45.8	97.3	59.9	47.0	88.9	105.6	57.7	42.9	33.0	14.5
	Cases 11-20	140.6	15.4	4.9	75.3	58.1	38.5	130.1	5.1	5.4	169.8	69.1	96.2	72.4	21.4	11.7
	Cases 01-20	130.4	22.8	6.9	74.3	59.0	42.0	112.5	17.4	16.0	122.8	85.4	74.5	55.7	26.6	13.0
$\times$10-*gen*	Cases 01-10	36.3	22.0	12.2	64,0	47,6	34,7	71.2	8.0	11.1	136.5	122.6	121.7	56.2	45.8	35.7
	Cases 11-20	45.4	8.4	4.2	74,5	55,2	42,1	57.0	0.6	0.7	189.7	99.1	160.4	56.9	19.5	15.6
	Cases 01-20	40.6	13.6	7.1	69,1	51,3	38,3	63.7	2.3	2.9	160.9	110.2	139.7	56.6	29.9	23.6

clearer performance dominance of *3bin* over *1bin* was observed when the experiments were carried out without the exponential startup cost constraints (for the sake of brevity, these results are not shown here but the interested reader is referred to [25]).

2) *Overall performance of P2:* For *1bin*, all the large cases (gray area in Fig. 2) took longer than one hour to solve within the required optimality tolerance. Two of the instances hit the ten hour time limit. On the other hand, *P2* solved all the tests in less than one hour and just two of them took more than 20 minutes. *P2* presented shorter runtimes than *1bin* and *3bin* for all the instances, being beaten just once by *P1* (instance 17 for the $\times$7-*day* test-set, see Fig. 2a). Note in Figs. 2a and 2c that for the instances where the proposed formulation showed the worst performances, *P2* required 57.1% and 19.5% of the CPU time required by *1bin* for the $\times$7-*day* and $\times$10-*gen* test-sets respectively. Furthermore, *P2* obtained better solution quality in general (i.e., converged to smaller optimality tolerances) than *1bin*, especially for the large cases. However, *1bin* obtained better solution qualities for 5 of the 40 instances (3 for the $\times$7-*day* and 2 for the $\times$10-*gen*).

3) *Node enumeration vs. LP complexity:* Table V shows that for the $\times$10-*gen* test-set, all the formulations enumerated more nodes than *1bin* to find a solution within

Table VI: Overall Speedups

	3bin over *1bin*	*P1* over *1bin*	*P2* over *1bin*	*P1* over *3bin*	*P2* over *3bin*	*P2* over *P1*
Cases 01-10	1.5	3.7	9.2	2.4	6.1	2.5
Cases 11-20	1.3	8.8	22.1	7.0	17.7	2.5
Cases 01-20	1.4	5.7	14.3	4.1	10.4	2.5

the given optimality tolerance. As mentioned above, formulations with different integer variables lead to different enumeration tree and branching strategies. By default, an MILP solver's objective is to obtain satisfactory feasible solutions quickly using different strategies, such as branching, cuts and heuristics. The strategy that proves to be more effective is used more often (e.g. branching) and the others more seldom so that they do as little harm as possible [10]. That being said, larger enumerated nodes in one formulation do not mean longer solving times. In fact, exploring few nodes for long periods is an indication of the difficulty of the linear relaxed formulation [30]. Therefore, *P2* is able to explore more nodes than *1bin* in even shorter times mainly due to the compactness of the formulation. Focusing on the number of nodes explored can lead to misleading conclusions. That is why other works prefer to look at the number of simplex iterations rather than nodes in order to perform comparisons [15], [31].

Finally, Table VI presents the overall speedups of the formulations compared with each other. For a given instance, the speedup of, for example, *3bin* over *1bin* is obtained by dividing the runtime of *1bin* by the runtime of *3bin*, and the results are summarized using the geometric mean on the speedups of a group of instances. Both test-sets $\times$7-*day* and $\times$10-*gen* are grouped together and now the instances are separated into small, large and all cases (see Table II). In general, *3bin* is 1.4 times faster than *1bin* and presented a better performance for the small cases. *P2* was around 14 and 10 times faster than *1bin* and *3bin*, respectively. *P2* presented the best performance for the large cases, where *P2* was around 22 and 18 times faster than *1bin* and *3bin*, respectively. Note that *P1* already presents significant improvements over *1bin* (5.7 times faster) and *3bin* (4.1 times faster). In addition, *P2* is a further improvement on *P1*, being generally 2.5 times faster.

D. Difficulty of an MILP vs. its Number of Binary Variables

In this part, we present two sets of experiments to assess the impact on the convergence evolution and solving times due to the number of binary variables, and the simultaneous tight and compact characteristic of the proposed formulation. The UC is solved for the eight-generator power system by 1) pure branch-and-bound method (BB), for one to three days, and 2) the solver defaults, which is the complete branch-and-cut method including heuristics (BC+H), for three to five days.

The eight-generator data and power demand can be found in Appendix C. The spinning reserve requirement of 5% of the power demand has to be met for each hour; the non-served energy is not considered (i.e., the variable nse_t is removed from all formulations); and all UC problems are solved until they hit the time limit of one hour or until they reach optimality (more precisely to 10^{-6} of relative optimality tolerance).

Five different formulations are now considered. Three of the formulations described in Section III-A, *1bin*, *3bin* and *P2*. The other two formulations are the relaxed versions of *3bin* and *P2* and they are denoted as *R3bin* and *RP2*, respectively. Similarly to *1bin*, the commitment variable u_t is the only variable that is defined as binary for *R3bin* and *RP2*. That is, for *R3bin* and *RP2*, the set of variables v_{gt}, w_{gt} and δ_{gst} are defined as continuous variables within the interval $[0, 1]$. As discussed in Sections II-A2 and II-B3, once u_{gt} is defined as binary, the formulations *3bin* and *P2* allow relaxing the integrality condition of variables v_{gt}, w_{gt} and δ_{gst} because the constraints guarantee that these variables always take binary values.

The problem size for the different formulations, the optimal solutions and integrality gaps are shown in Table VII. The computational performances are presented in Table VIII. which includes runtimes, nodes explored, iterations and memory required to solve the problems until optimality, otherwise the final optimality tolerance is shown within parentheses. Table VII shows that all formulations present the same optimal solution for a given time span. As expected, all formulations obtain the same integer solutions because they are characterizing the same integer problem (see Section II).

1) Pure Branch and Bound (BB): To assess the impact on the computational performance of formulations containing different number of binary variables, the UC is solved by only using the branch-and-bound method. The cutting planes and heuristics were then disabled, and CPLEX defaults were used for all remaining features. Therefore, the solver is forced to explore all the tree in order to prove optimality [11], [16].

Table VIII shows the computational performance for the 1-, 2- and 3-day cases that where solved by pure BB. Table IX presents the optimal generation schedule, for the 1-day case, obtained by all formulations, of which schedules were the same. All formulations could prove optimality for the case of one and two days. For the 3-day case, none of the formulations could achieve optimality because they either exceeded the one-hour time limit or the 4-GB memory limit. *P2*, *3bin*, *RP2* and *R3bin* hit the time limit and the final optimality tolerances that they achieved are shown between parentheses in Table VIII. *1bin* hit the memory limit achieving the worst final optimality tolerance of 4.3×10^{-3} after about 500 seconds.

Although *P2* is the formulation with the highest number of binary variables, *P2* explored the least number of nodes and presented the best computational performance for the 2-day case, in comparison with the other five formulations. On the other extreme, *1bin*, with the least number of binary variables (the same as *R3bin* and *RP2*), explored the highest number of nodes and presented the worst computational performance, followed by *R3bin* and *RP2*, see Table VIII.

Fig. 3 shows the convergence evolution until optimality of *P2*, *3bin* and *1bin*, for the 2-day case. Note that the first value of the lower bound of *P2*, which is the LP relaxed solution, was found sooner (due to the compactness) and nearer to the final integer solution (due to the tightness) than *3bin* and *1bin*. After 4 seconds, *P2* presented a better evolution of both lower

and upper bounds and hence faster convergence to optimality in comparison with *3bin* and *1bin*.

If a model containing n binary variables is solved by the pure BB algorithm, this algorithm could potentially enumerate 2^n nodes to prove optimality. Consequently, one might expect the solution time to increase exponentially with the number of binary variables. However, the BB method cuts off large sections of the potential tree because some expected solutions may be infeasible or worse than solutions already known. For example, for the 2-day case, *P2* potentially presents $2^{1920-384}$ times more nodes to explore than *1bin*, but the BB algorithm solved *P2* until optimality enumerating 42% of the number of nodes required to solve *1bin*. This very surprising efficiency that the BB method exhibits over the potential amount of computation is due to the tightness of the formulation, hence the number of binary variables is a very poor indicator of the difficulty of an MILP model [11], [16]. In fact, the BB will solve a problem exploring zero nodes if the whole model is the tightest possible (convex hull); that is, the LP relaxation solution will always be integer [11]. Furthermore, increasing the number of binary variables is actually a tightening strategy [32].

Increasing the number of binary variables can also provide a further advantage in the tree search strategy. This is the case of formulations involving variables that take binary values even when these variables are defined as continuous, e.g., v_{gt}, w_{gt} and δ_{gst}. As also stated in [19], declaring all these variables as binary does not cause extra complexity during the enumeration process (branching), because when fixing some of the variables, many others can be immediately obtained due to the high correlation among each other. For example, if the variable v_{gt} is fixed to 1 then (6) fixes u_{gt} to 1 for the following TU_g periods, then (8) (together with (7)) fixes w_{gt} and v_{gt} to 0 for those TU_g periods, and finally (3) fixes δ_{gst}, for all s, for the same periods.

2) Branch-and-Cut + Heuristics (BC+H): The UC for the 8-generator system was solved for three, four and five days using BC+H strategy, which are CPLEX defaults. Similarly, as previously shown in Section III-D1, *P2* generally requires shorter runtimes to solve the problem until optimality than the other four formulations, see Table VIII. *P2* was up to 17 and 2.5 times faster than *1bin* and *3bin*, respectively. Note that, in general, the relaxed versions *R3bin* and *RP2* present a higher computational burden than their analogous formulations *3bin* and *P2*, respectively.

Note that in Table VIII, the formulations with higher number of binary variables may keep the tree considerably smaller in size. For example, for the 5-day case, *1bin* required around 250 and 110 times more memory to deal with the branch-and-bound tree than *P2* and *3bin*, respectively. In order to solve the 5-day case until optimality, *P2* potentially presents $2^{4800-960}$ times more nodes to explore than *1bin*, but the BB+H algorithm solved *P2* and *3bin* by just enumerating around 3% of the number of nodes required to solve *1bin*. Similarly, *3bin* required around the 5% of the nodes enumerated by *1bin*, although *3bin* potentially presented $2^{2880-960}$ times more nodes to explore. As mentioned before, the tightness and the high correlation between binary variables of *3bin* and *P2* considerably reduce the size of the tree.

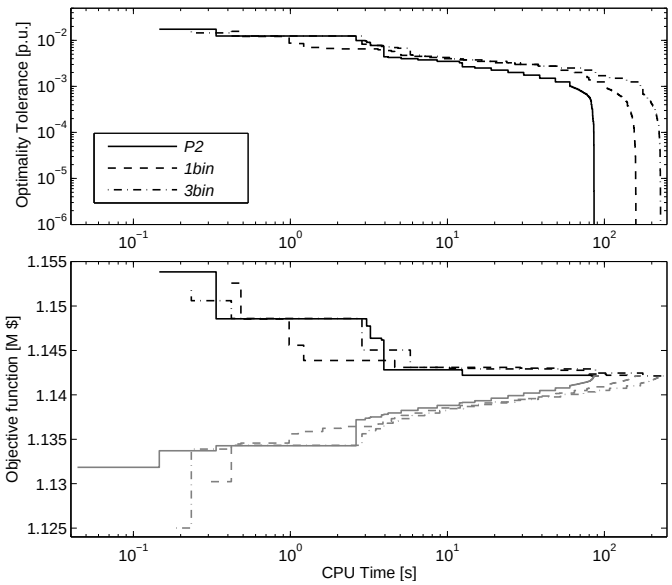


Fig. 3: Convergence evolution for the optimality tolerance and upper/lower bounds in the up and bottom part of the figure, respectively. In the bottom figure, black lines refer to upper bounds and gray lines to lower bounds.

In short, the computational burden of a MILP formulation mainly depends on its tightness and compactness. A further computational advantage can be obtained by defining as binary those variables that either way take binary values, because the constraints (and tightness of the model) force them to do so. In addition, branching on different variables may be more convenient [10], [19]. MILP solvers employ techniques to exploit the integrality characteristic of integer variables such as cutting planes and node presolve [10], [11], [20]. Consequently, declaring this type of variables as continuous will not allow the solver to look for opportunities to exploit their integrality characteristic.

IV. Conclusions

This paper presented an MILP reformulation for the thermal UC problem. The formulation is simultaneously tighter and more compact than equivalent formulations found in the literature. This simultaneous characteristic boosts the convergence speed that solvers take to solve the MILP formulation. This is done by reducing the search space (tightening) and at the same time increasing the searching speed (compacting) with which solvers explore that reduced space. Consequently, the computation time is dramatically reduced. Several case studies were analysed to show the improvements achieved by this formulation with respect to others available in the literature. Results showed that the proposed UC formulation considerably reduced the computational burden while achieving better solution qualities. While the formulation is tested only on "standard" thermal UC problems, the tight and compact formulation can be further extended to many other variants of the UC problem, where analogous results should be expected.

Table VII: Problem Size, Optimum and Integrality Gap for the 8-generator Case

days	# constraints			# nonzero elements			# binary var			# total var			Optimum	Integrality Gap [$\times 10^{-3}$]		
	*P2**	*3bin*†	*1bin*	*P2**	*3bin*†	*1bin*	*P2**	*3bin*†	*1bin*‡	*P2**	*3bin*†	*1bin*	*All*$^{\diamond}$	*P2**	*3bin*†	*1bin*
1	1480	4587	4647	7105	30103	29897	960	576	192	1344	1152	960	573630.655	10.21	12.07	17.86
2	3088	9243	9303	15097	66151	65513	1920	1152	384	2688	2304	1920	1142132.128	9.01	10.43	15.00
3	4696	13899	13959	23089	102199	101129	2880	1728	576	4032	3456	2880	1710633.601	8.61	9.88	14.04
4	6304	18555	18615	31081	138247	136745	3840	2304	768	5376	4608	3840	2279135.074	8.41	9.60	13.55
5	7912	23211	23271	39073	174295	172361	4800	2880	960	6720	5760	4800	2847636.547	8.29	9.43	13.27

* *P2* is equal to *RP2* for these cases
† *3bin* is equal to *R3bin* for these cases
‡ *1bin*, *RP2* and *R3bin* are equal for these cases
$^{\diamond}$ *P2*, *3bin*, *1bin*, *RP2* and *R3bin* are equal for these cases

Table VIII: Computational Performance for the 8-generator Case

	days	Time [s] / (OptTol [$\times 10^{-3}$])					Nodes [$\times 10^{3}$]					Iterations [$\times 10^{3}$]					Memory [MB] / (OptTol [$\times 10^{-3}$])				
		P2	*3bin*	*1bin*	*RP2*	*R3bin*	*P2*	*3bin*	*1bin*	*RP2*	*R3bin*	*P2*	*3bin*	*1bin*	*RP2*	*R3bin*	*P2*	*3bin*	*1bin*	*RP2*	*R3bin*
BB	1	1.1	0.6	0.3	0.5	1.1	1.8	0.9	0.6	1.2	1.4	15.6	18.1	9.6	12.3	30.2	0.1	0.1	0.1	0.1	0.1
	2	92.0	169.8	242.7	171.5	205.5	92.4	170.6	220.5	232.0	130.6	1278.2	2140.0	3952.9	3082.2	2733.2	23.0	46.9	48.1	44.9	373.5
	3	(2.03)	(1.04)	518.3	(3.12)	(2.79)	3150.8	2692.7	248.7	3569.8	1552.3	49814.7	47095.2	5368.0	62240.6	51563.5	1349.2	1010.8	(4.3)	1079.2	1313.6
BC+H	3	7.9	9.9	4.3	7.5	6.6	1.1	1.1	0.8	0.7	0.8	38.6	80.0	37.5	27.8	52.4	1.2	0.2	0.3	0.3	1.4
	4	9.7	22.4	44.0	12.4	277.0	0.6	1.0	6.4	1.1	27.8	25.6	128.4	296.5	43.1	1884.8	0.7	1.7	4.4	5.5	7.8
	5	12.9	33.4	218.7	39.2	1594.5	0.9	1.4	29.6	4.8	220.6	34.5	158.9	1157.3	172.1	11339.5	1.0	2.3	247.8	1.9	128.2

Table IX: Optimal Generation Scheduling for the 8-generator Case and 1-day Time Span [MW]

Gen	Hour																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	375	455	455	455	455	455	455	455	455	455	455	455	455	455	455	455	455	455	455	455	455	455	455	455
2	375	418.45	455	451.2	420.16	420.16	443.26	418.67	444.28	455	455	455	455	455	453.7	433.51	426.16	455	455	455	455	455	437.4	408.48
3	70	20	.	.	.	.	.	20	70	120	130	130	130	130	130	130	130	130	130	130	130	130	130	130
4	70	20	.	.	.	.	.	20	70	120	130	130	130	130	130	130	130	130	130	130	130	130	130	130
5	85	40.35	52.24	25	25	25	32.94	79.61	93.68	91.6	82.64	98.16	82.64	71.6	57.38	77.57	102	162	162	150.76	129.2	113.68	53.68	25
6	80	20	.	.	.	.	.	.	.	.	20	20	20	.	.	.	20	55.32	39.8	20	20	20	20	.
7	25	25	.	.	.	.	.	.	.	.	.	.	.	.	.	.	25	25	25	25	.	.	.	.
8	21.92	10	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

ACKNOWLEDGEMENT

Germán Morales-España has been awarded an Erasmus Mundus Ph.D. Fellowship. The authors would like to express their gratitude to all partner institutions within the programme as well as the European Commission for their support.

APPENDIX

A. Initial Conditions

The initial behaviour of the units is bound by their initial conditions. This is guaranteed by fixing the value of some variables before running the optimization model. The following parameters are needed to deal with the unit state during the first periods:

- u_g^0 Initial commitment status of unit g $\{0,1\}$.
- TU_g^0 Number of hours that the unit g has been online before the scheduling horizon.
- TD_g^0 Number of hours that the unit g has been offline before the scheduling horizon.

a) Initial Minimum Up/Down Times: The number of hours during which the units must be initially online (TU_g^R) or offline (TD_g^R) due to their minimum up/down constraints are obtained as follows:

$$TU_g^R = \max\left\{0, \left(TU_g - TU_g^0\right) u_g^0\right\} \qquad \forall g \quad (13a)$$

$$TD_g^R = \max\left\{0, \left(TD_g - TD_g^0\right) \left(1 - u_g^0\right)\right\} \qquad \forall g \quad (13b)$$

Now, the commitment variables for the initial periods where the units must remain online or offline $\left(TU_g^R + TD_g^R\right) \geq 1$ must be fixed:

$$u_{gt} = u_g^0 \qquad \forall g, t \in \left[1, TU_g^R + TD_g^R\right] \quad (14)$$

b) Initial Startup Type: The equation (15) complements (2) and this means that an inappropriate startup type will not be chosen when taking into account the initial conditions. If $TD_g^0 \geq 2$, then δ_{gst} must be fixed for some initial periods as follows:

$$\delta_{gst} = 0 \qquad \forall g, s \in [1, S_g)\,,\, t \in \left(T_{g,s+1}^{SU} - TD_g^0, T_{g,s+1}^{SU}\right). \quad (15)$$

Equations (14) and (15) can also be respectively written as additional constraints in the optimization problem:

$$\sum_{t=1}^{TU_g^R + TD_g^R} u_{gt} = TU_g^R u_g^0 \qquad \forall g \quad (16)$$

$$\sum_{s=1}^{S_g - 1} \sum_{t=T_{g,s+1}^{SU} - TD_g^0 + 1}^{T_{g,s+1}^{SU} - 1} \delta_{gst} = 0 \qquad \forall g. \quad (17)$$

However, equations (14) and (15) are preferred over (16) and (17), because (14) and (15) turn the involved integer variables into constants before running the optimization problem, thus decreasing the problem size. Nevertheless, solvers usually provide the option to treat fixed variables as constants.

B. Comparing Feasible Regions of the Startup Capability Constraints

This section compares the tightness of the startup capability constraint of the proposed formulation with the one presented

in [21]. Similar analysis and conclusions can be made for the shutdown capability constraint. For notational simplicity, the index g is dropped in this section.

For all the following analysis, we consider $u_{t-1} = 0$. Therefore, the constraint imposing the unit's startup capability (9) becomes

$$p_t + r_t \leq \left(\overline{P} - \underline{P}\right) u_t - \left(\overline{P} - SU\right) v_t \qquad \forall t \quad (18)$$

and (6) together with 8 imposes $v_t = u_t$. That is, (6) guarantees $v_t \leq u_t$, and if $u_{t-1} = 0$ then (8) ensures $v_t \geq u_t$, thus forcing $v_t = u_t$. Consequently, (18) can be rewritten as a function of u_t:

$$p_t + r_t \leq (SU - \underline{P})\, u_t \qquad \forall t. \quad (19)$$

The analogous constraint from [21, eq. (18)], imposing the unit's startup capability, becomes

$$\overline{p}_t \leq SU \cdot u_t + \overline{P}\,(1 - u_t) \qquad \forall t \quad (20)$$

when $u_{t-1} = 0$. Where $\overline{p}_t$ is the maximum available power output at time t, which is the total power output plus the spinning reserve. Thus, in terms of the nomenclature used in this paper, $\overline{p}_t$ is equal to $\underline{P} \cdot u_t + p_t + r_t$. Hence, (20) can be rewritten as

$$p_t + r_t \leq (SU - \underline{P})\, u_t + \overline{P}\,(1 - u_t) \qquad \forall t. \quad (21)$$

Now the tightness of inequalities (19) and (21) can be directly compared. Note that if the unit starts up, $u_t = 1$, both (19) and (21) impose $p_t + r_t \leq SU - \underline{P}$. Therefore, the feasible integer region of both constraints is the same when the unit starts up. Be aware, however, that their relaxed feasible regions are completely different. Whereas the right side of (19) takes its maximum value when $u_t = 1$, the right side of (21) actually takes its minimum value. That is, the term involving $\overline{P}$ in (21) plays the role of the so-called big-M, so that (21) becomes inactive when $u_t = 1$. The big-M inequalities considerably harm the tightness of MILP formulations so they must be avoided when possible [12], [16], [18].

Furthermore, if $SU = \underline{P}$, which is a very common case [19], [21], [33], then (19) and (21) respectively become

$$p_t + r_t \leq 0 \qquad \forall t \quad (22)$$

$$p_t + r_t \leq \overline{P}\,(1 - u_t) \qquad \forall t. \quad (23)$$

Although these two constraints impose $p_t + r_t \leq 0$, when the unit starts up $u_t = 1$, the right side of (23) provides a very poor upper bound to p_t and r_t when $u_t \in [0, 1)$.

C. Power System Data

The eight-unit system data used in [21] and [19] are shown in Table X and the hourly demand (used in [19]) depending on the power system's total capacity is shown in Table XI.

The startup and shutdown rates are assumed equal to the unit minimum output ($SU_g = SD_g = \underline{P}_g$). For the numerical experiments in Section III-C, the initial power production of units 1 and 2, prior to the first period of the time span, is 455MW and 245MW respectively. For the experiments in Section III-D, the initial power production of all units is their minimum output $\underline{P}$, hence the initial states Ste_0 in Table X are considered the same in magnitude but positive for all units.

Table X: Generator Data

	Technical Information						Cost Coefficients			
Gen	$\overline{P}$ [MW]	$\underline{P}$ [MW]	TU/TD [h]	RU/RD [MW/h]	Ste_0* [h]	T_c^{SU} [h]	C^{NL} [$/h]	C^{LV} [$/MWh]	$C_h^{SU\dagger}$ [$]	$C_c^{SU\dagger}$ [$]
1	455	150	8	225	8	14	1000	16.19	4500	9000
2	455	150	8	225	8	14	970	17.26	5000	10000
3	130	20	5	50	-5	10	700	16.60	550	1100
4	130	20	5	50	-5	10	680	16.50	560	1120
5	162	25	6	60	-6	11	450	19.70	900	1800
6	80	20	3	60	-3	8	370	22.26	170	340
7	85	25	3	60	-3	6	480	27.74	260	520
8	55	10	1	135	-1	2	660	25.92	30	60

*Hours that the unit has been online (+) or offline (-) prior to the first period of the time span.

†Subindex h refers to hot startup $s = 1$, and subindex c to cold startup $s = 2$

Table XI: Demand (% of Total Capacity)

Time	1	2	3	4	5	6	7	8	9	10	11	12
Demand	71%	65%	62%	60%	58%	58%	60%	64%	73%	80%	82%	83%
Time	13	14	15	16	17	18	19	20	21	22	23	24
Demand	82%	80%	79%	79%	83%	91%	90%	88%	85%	84%	79%	74%

REFERENCES

[1] S. Stoft, *Power System Economics: Designing Markets for Electricity*, 1st ed. Wiley-IEEE Press, May 2002.

[2] B. F. Hobbs, W. R. Stewart, R. E. Bixby, M. H. Rothkopf, R. P. O'Neill, and H.-p. Chao, "Why this book? new capabilities and new needs for unit commitment modeling," in *The Next Generation of Electric Power Unit Commitment Models*, B. F. Hobbs, M. H. Rothkopf, R. P. O'Neill, and H.-p. Chao, Eds. Boston: Kluwer Academic Publishers, 2002, vol. 36, pp. 1–14.

[3] N. Padhy, "Unit commitment-a bibliographical survey," *IEEE Transactions on Power Systems*, vol. 19, no. 2, pp. 1196–1205, May 2004.

[4] 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, and K. Wolter, "MIPLIB 2010," *Mathematical Programming Computation*, vol. 3, no. 2, pp. 103–163, Jun. 2011.

[5] A. L. Ott, "Evolution of computing requirements in the PJM market: Past and future," in *2010 IEEE Power and Energy Society General Meeting*. IEEE, Jul. 2010, pp. 1–4.

[6] T. Li and M. Shahidehpour, "Price-based unit commitment: a case of lagrangian relaxation versus mixed integer programming," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 2015–2025, Nov. 2005.

[7] F. Bouffard, F. Galiana, and A. Conejo, "Market-clearing with stochastic security-part i: formulation," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1818–1826, Nov. 2005.

[8] G. Morales-España, A. Ramos, and J. García-González, "An MIP formulation for joint market-clearing of energy and reserves including ramp scheduling," *IEEE Transactions on Power Systems*, 2013, paper under Review (Manuscrit ID: TPWRS-00510-2012.R2), online preprint. [Online]. Available: http://www.iit.upcomillas.es/~aramos/papers/V3.4_UC-based_MC.pdf

[9] K. Hedman, M. Ferris, R. O'Neill, E. Fisher, and S. Oren, "Co-optimization of generation unit commitment and transmission switching with n-1 reliability," *Power Systems, IEEE Transactions on*, vol. 25, no. 2, pp. 1052 –1063, May 2010.

[10] R. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling, "MIP: theory and practice–closing the gap," in *System Modelling and Optimization: Methods, Theory and Applications*, M. J. D. Powell and S. Scholtes, Eds. Boston: Kluwer Academic Publishers, 2000, vol. 174, p. 19–49.

[11] L. Wolsey, *Integer Programming*. Wiley-Interscience, 1998.

[12] F. Vanderbeck and L. A. Wolsey, "Reformulation and decomposition of integer programs," in *50 Years of Integer Programming 1958-2008*, M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 431–502.

[13] E. Rothberg, "The CPLEX library: Presolve and cutting planes," in *4th Max-Planck Advanced Course on the Foundations of Computer Science (ADFOCS)*, Saarbrücken, Germany, Sep. 2003, pp. 1–17.

[14] R. Bixby and E. Rothberg, "Progress in computational mixed integer programming—A look back from the other side of the tipping point," *Annals of Operations Research*, vol. 149, no. 1, pp. 37–41, Jan. 2007.

[15] A. Lodi, "Mixed integer programming computation," in *50 Years of Integer Programming 1958-2008*, M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 619–645.
[16] H. P. Williams, *Model Building in Mathematical Programming 4th Edition*, 4th ed. John Wiley & Sons, Aug. 1999.
[17] R. E. Bixby, "Solving real-world linear programs: A decade and more of progress," *Operations Research*, vol. 50, no. 1, pp. 3–15, Jan. 2002.
[18] J. Hooker, *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*, 1st ed. Wiley-Interscience, May 2000.
[19] J. Ostrowski, M. F. Anjos, and A. Vannelli, "Tight mixed integer linear programming formulations for the unit commitment problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 39–46, Feb. 2012.
[20] L. Wolsey, "Strong formulations for mixed integer programs: valid inequalities and extended formulations," *Mathematical programming*, vol. 97, no. 1, p. 423–447, 2003.
[21] M. Carrion and J. Arroyo, "A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem," *Power Systems, IEEE Transactions on*, vol. 21, no. 3, pp. 1371–1378, 2006.
[22] D. Rajan and S. Takriti, "Minimum Up/Down polytopes of the unit commitment problem with start-up costs," IBM, Research Report RC23628, Jun. 2005. [Online]. Available: http://domino.research.ibm.com/library/cyberdig.nsf/1e4115aea78b6e7c85256b360066f0d4/cdcb02a7c809d89e8525702300502ac0?OpenDocument
[23] A. Frangioni, C. Gentile, and F. Lacalandra, "Tighter approximated MILP formulations for unit commitment problems," *IEEE Transactions on Power Systems*, vol. 24, no. 1, pp. 105–113, Feb. 2009.
[24] G. Morales-Espana, J. M. Latorre, and A. Ramos, "Tight and compact MILP formulation of start-up and shut-down ramping in unit commitment," *IEEE Transactions on Power Systems*, vol. PP, no. 99, p. 1, 2012, in press.
[25] G. Morales-España, J. M. Latorre, and A. Ramos, "Online companion for tight and compact MILP formulation for the thermal unit commitment problem," Research Report IIT-12-072, 2012. [Online]. Available: http://www.iit.upcomillas.es/~aramos/papers/OnlineCompanion_Tight&Compact_UC.pdf
[26] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation, and Control*, 2nd ed. Wiley-Interscience, Jan. 1996.
[27] M. P. Nowak and W. Römisch, "Stochastic lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty," *Annals of Operations Research*, vol. 100, no. 1, pp. 251–272–272, Dec. 2000.
[28] K. Hedman, R. O'Neill, and S. Oren, "Analyzing valid inequalities of the generation unit commitment problem," in *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, 2009, pp. 1–6.
[29] "The GAMS development corporation website," 2012, www.gams.com.
[30] R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling, "Mixed-integer programming: A progress report," in *The Sharpest Cut: The Impact of Manfred Padberg and his Work*, ser. MPS-SIAM Series on Optimization, M. Grötschel, Ed. 3600 Market Street, 6th Floor Philadelphia, PA 19104-2688: SIAM, Jan. 2004, pp. 309—325.
[31] E. Danna, "Performance variability in mixed integer programming," in *5th Workshop on Mixed Integer programming*, 2008, pp. 1–40.
[32] H. P. Williams, *Logic and integer programming*, 1st ed., ser. International series in operations research & management science. New York: Springer, 2009, no. v. 130.
[33] L. Wu, M. Shahidehpour, and T. Li, "Stochastic security-constrained unit commitment," *Power Systems, IEEE Transactions on*, vol. 22, no. 2, pp. 800 –811, May 2007.

Germán Morales-España (S' 2010) received the B.Sc. degree in electrical engineering from the Universidad Industrial de Santander, Bucaramanga, Colombia, in 2007 and the M.Sc. degree from the Delft University of Technology, Delft, The Netherlands, in 2010. He is now pursuing the (Ph.D.) Erasmus Mundus Joint Doctorate in Sustainable Energy Technologies and Strategies (SETS) hosted by the Universidad Pontificia Comillas, Madrid, Spain; the Royal Institute of Technology, Stockholm, Sweden; and Delft University of Technology, Delft, The Netherlands.

He is currently an assistant researcher at the Institute for Research in Technology (IIT) at the Universidad Pontificia Comillas, and he is also a member of the Research Group on Electric Power Systems (GISEL) at the Universidad Industrial de Santander. His areas of interest are power systems operation, economics and reliability, as well as power quality and protective relaying.

Jesus M. Latorre (S'00-M'07) was born in Madrid, Spain, in 1977. He received the degree of Electronic Engineer in 2001 and the Ph.D. degree in November 2007, from Comillas Pontifical University, Madrid, Spain.

He is currently a postdoctoral researcher at the Institute for Research in Technology, of the Comillas Pontifical University. His main interest areas include operations research and mathematical modelling, stochastic programming, parallel and distributed computing, algorithms and numerical methods.

Andres Ramos received the degree of Electrical Engineering from Universidad Pontificia Comillas, Madrid, Spain, in 1982 and the Ph.D. degree in Electrical Engineering from Universidad Politécnica de Madrid, Madrid, Spain, in 1990.

He is a Research Fellow at Instituto de Investigación Tecnológica, Madrid, Spain, and a Full Professor at Comillas' School of Engineering, Madrid, Spain, where he has been the Head of the Department of Industrial Organization. His areas of interest include the operation, planning, and economy of power systems and the application of operations research to industrial organization.