ILP resolution of Unit Commitment problem with minimum stop constraints

N. Dupin$^{1,2}$ and P. Bendotti$^2$

1. IMB, Université Bordeaux 1, 351 cours de la libération, 33405 Talence Cedex
Nicolas.Dupin@math.u-bordeaux1.fr

2. EDF-R& D, département OSIRIS, 1 avenue du Général de Gaulle, 92140 Clamart, France
pascale.bendotti@edf.fr

Keywords : Unit Commitment Problem, power production scheduling, Integer Linear Programming, Polyhedrons, Constraint reformulation, Cut generation.

1 UCPd, a discrete Unit Commitment Problem

The Unit Commitment Problem (UCP) consists in finding a power generators scheduling satisfying a demand while minimizing the total production cost over the considered time horizon. Since electricity is not storable, the total electricity production of all units has to meet the demand at every time step.

In the short term UCP considered in this paper, the scheduling is based on a forecasted demand. Safety capacity production is required for real-time balance of production and demand. Therefore, the scheduling must fulfill a reserve demand, similar to the power demand.

In this paper, a special case of UCP is considered, where all plants must operate for a minimum time on different levels of power, called operating points. This way of operation defines UCPd, the Unit Commitment Problem with discrete operation stops. Hence, at every time step, each demand can be satisfied as with a sum of discrete operating points. This underlying knapsack structure proves that UCPd is NP-Hard.

In the actual daily generation scheduling problem at Électricité de France (EDF), fossil power plants are modelled like in UCPd. Actual resolution method is based on price decomposition, where coupling constraints on demands are dualized and subproblems on each fossil power plant are solved using dynamic programming (see [5] for more details).

In this paper, UCPd is modelled as an Integer Linear Program (ILP). We will develop and improve compact formulations to fit with a convenient use of modern ILP solvers. Formulations are implemented with Cplex solver, with characteristics described in [7].

2 First ILP compact modeling : state formulation

We model UCPd only with binary variables. In particular, state variable $s_{u,t}^{(i)}$ and Generalized Upper Bound (GUB) constraints : $\sum_i s_{u,t}^{(i)} \leq 1 \ \forall u, \forall t$ are used to indicate if unit $u$ produces at time step $t$ on point $i$. Efficient branching is provided with Special Ordered Sets (SOS) implementation. Indeed, Branch&Bound trees with SOS are well-balanced and require a linear number of nodes, while standard branching on variables (see [3]) leads to trees with an exponential number of nodes.

Demands on power and reserve are knapsack constraints similar to $\sum_{u,i} \alpha_{u,t}^{(i)} s_{u,t}^{(i)} \geq d_t, \alpha_{u,t}^{(i)} \geq 0 \ \forall t$. Cover cuts will naturally be generated for these constraints. Furthermore, these cuts can be strengthened with GUB constraints defining $s_{u,t}^{(i)}$. It leads to GUB Cover cuts, as described in [2].
Adding start up variables to each operating point, we can write dynamic constraints with enforced minimum stop generalizing [1]. This constraints formulation has good polyhedral properties.

Solvers like Cplex can not take profit simultaneously of GUB Cover cuts generation and SOS implementation, see [7]. It is the weak spot of state formulation, for an efficient use with Cplex.

**3 Polyhedral transformation, level formulation**

Let us consider a new formulation, referred as level formulation, where level variables \( l_{u,t}^{(i)} \) are defined to 1 if the generating power is greater than the power level \( i \). This leads to modeling constraints \( l_{u,t}^{(i)} \leq l_{u,t}^{(i+1)} \), instead of previous GUB constraints. Standard branching on variables \( l_{u,t}^{(i)} \) is equivalent to efficient SOS branching with variables \( s_{u,t}^{(i)} \).

![Figure 1 – State variables](image1.png)

![Figure 2 – Level variables](image2.png)

Constraints considered for UCPd in state formulation can easily be rewritten with the latter variables. In the demand constraints, the coefficient associated to a variable \( l_{u,t}^{(i)} \) is the power difference to the previous level. Power demands are always knapsack constraints as powers increase with \( i \). This does not hold in general for reserve demands, as reserve capacities do not increase with \( i \).

\[
\varphi : (x_1, \ldots, x_j, \ldots, x_n) \mapsto \left( \sum_{i=1}^{n} x_i, \ldots, \sum_{i=j}^{n} x_i, \ldots, x_n \right)
\]

defines an isomorphism that transforms state variables to level variables. Let us define an isomorphism \( \Phi \) extending \( \varphi \) on each variable \( s_{u,t}^{(i)} \). Hence, \( \Phi \) is a bijective correspondence between the integer points of both state and level formulations.

Image by \( \Phi \) of hyper-planes \( \sum_{u,t,i} a_{u,t,i}^{(i)} s_{u,t}^{(i)} \leq d \) are hyper-planes \( \sum_{u,t,i} (a_{u,t,i}^{(i)} - a_{u,t,i}^{(i-1)}) l_{u,t}^{(i)} \leq d \) for all vector family \( a_{u,t,i}^{(i)} \) with extension \( a_{u,t,i}^{(0)} = 0 \).

Using \( \Phi \) or \( \Phi^{-1} \) projects the constraints from one formulation to the other, leading to polyhedrally equivalent formulations. Furthermore, it allows us to compare the projected linear description with the original one. Both state and level formulations can be improved this way. We get equivalent improved formulations in terms of polyhedral description, and thus in terms of LP relaxation.

**4 Implementation results**

Even though both improved state and level formulations are theoretically equivalent, there are always differences in their ILP resolution. In particular, we already mentioned that branching on level variables is more efficient that branching on state variables.
Heuristics implemented in ILP solvers like RINS or Feasibility Pump (see [4] for a survey) solve small ILP by fixing integer variables. Fixing some integer level variables induces more flexible decisions than fixing state variables. In our implementation, primal bound convergence was greatly improved with level formulation, whereas it was slow with state formulation.

On both formulations, main cuts generated by Cplex belong to the classes of cover cuts and specific rounding cuts, “zero half cuts”. At every time step, demand constraints in power and reserve share the same variables. Therefore, zero half cut can be generated.

A question that arises now is whether Cplex generates equivalent cuts in both formulations. Before branching, cuts were in general much more efficient with the state formulation. It led us to experiment a mixed formulation with both sets of variables $s^{(i)}_{u,t}, l^{(i)}_{u,t}$ adding the linking constraint $\varphi(s_{u,t}) = l_{u,t}$, and the constraints implying a special structure detection. Since it improved significantly the previous dual bounds, it tends to show that both sets of cuts are not equivalent.

We can take profit of both formulations without increasing the number of variables. The idea is to generate Cplex cuts at the root node of the state formulation and to project these cuts using $\Phi$ hyper-plane transformation. The ILP finally solved is based from the level formulation, thus still benefiting from original cuts generated by cplex, and includes cuts obtained through projection from the state formulation. This resolution relying on one set of variables will take advantage of efficient branching and heuristics, thus leading to improved dual bound at the root node before branching.

**Conclusion and perspective**

We elaborated on two compact ILP formulations for UCPd. Both formulations were improved with polyhedral transformations, leading to polyhedrally equivalent formulations. For an effective use of modern ILP solvers, detection of specific structures is decisive for the integer resolution.

Computing time is mainly devoted to guarantee the optimality of the solution. For industrial use, it is relevant to trade optimality for computing time, especially when quality of solutions and time saving are both high. A perspective is to accelerate the convergence of our primal bounds. On the one hand, the quality of our LP relaxation can be used to guide a local search. On the other hand, our work on ILP resolution can help to implement ILP neighborhoods within a parallel Variable Neighborhood Search (VNS) scheme.

**Références**


