Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

Solving Realistic Unit Commitment Problems Using Temporal Planning: Challenges and Solutions

Chiara Piacentini, Daniele Magazzeni, Derek Long, Maria Fox
Department of Informatics, King's College London, UK
name.surname@kcl.ac.uk

Chris Dent
School of Engineering and Computing Sciences
University of Durham, UK
chris.dent@durham.ac.uk

Abstract

When facing real world planning problems, standard planners are often inadequate and enhancement of the current techniques are required. In this paper we present the challenges that we have faced in solving the Unit Commitment (UC) problem, a well-known problem in the electrical power industry for which current best methods are based on Mixed Integer Programming (MIP). Typical UC instances involve hundreds or even thousands of generating units, pushing the scalability of state of the art planners beyond their limits. Furthermore, UC is characterised by state-dependent action costs, a feature that not many domain independent planners can efficiently handle. In this paper we focus on the challenge of making domain-independent planning competitive with the MIP method on realistic-sized UC instances. We present the results of our investigation into modelling the UC problem as a temporal planning problem, and show how we scaled up from handling fewer than 10 generating units to more than 400, obtaining solutions almost as high quality as those generated by MIP. We conclude by discussing future directions for temporal planning in this domain, that lie beyond what can be modelled and solved using MIP methods.

1 Introduction

In power systems engineering, the Unit Commitment (UC) is the problem of finding which generating units to switch on or off and when, so that a forecasted demand is satisfied. Once a set of committed units is given, the sub-problem of determining the output of each unit is called Economic Dispatch (ED) and it is solved alongside the UC problem in order to minimise the total cost of production of the electricity (Wood, Wollenberg, and Sheblé 2013).

The UC problem has attracted the interest of both academia and industry over many decades, since it has a significant economic impact when managing a power system. The current *state-of-the-art* techniques for solving Unit Commitment are based on Mixed Integer Programming methods (MIP), for which efficient commercial solvers are available. MIP is restricted to linear constraint modelling (although a quadratic objective function can be used), which limits the scope for extending MIP methods to consider richer models of power system problems. The interest in studying the UC problem has recently been growing because of a global focus on carbon emissions and the potential for increasing the role of renewable resources in the production of electricity. The efficient use of wind farms and PV installations, and the role of storage which introduces non-linear constraints, poses many new challenges for modelling and solving problems like Unit Commitment. This motivates an interest in finding alternative techniques that can model more complex constraints and scale up to consider more complex scenarios.

As argued by Campion et al. (2013), AI Planning presents some potential benefits over the MIP formulation. A first advantage of AI Planning over MIP is in the modelling of the problem. A MIP formulation relies on a fixed discretisation of the timeline and the duplication of all the variables for each time-point considered. AI Planning offers a more compact model, expressed in terms of applicable actions using one of the available planning languages. In particular we consider the numeric and temporal extensions of PDDL, such as PDDL2.1 (Fox and Long 2003) or PDDL+ (Fox and Long 2006), for which a number of domain-independent planners are available. Moreover, because of the inherent expressive power of these languages, non-linear constraints can be captured, and recent planning methods have proven capable of solving problems with such constraints (Bajada, Fox, and Long 2015; Cashmore et al. 2016). Furthermore, temporal planners such as POPF (Coles et al. 2010) and UPMurphi (Della Penna et al. 2009) treat time as continuous, and handle time-dependent changes. The treatment of time as a continuous variable avoids the need for a fixed discretisation, which can lead to a more accurate calculation and optimisation of the objective function. There is therefore reason to believe that temporal planning might offer a method that can surpass MIP when it comes to extending beyond the current linear models of problems like UC.

An advantage that MIP has over any kind of planning approach is scalability. While planning uses weak heuristic methods (distance to goal) based on the delete relaxation (Hoffmann and Nebel 2001), the MIP solution approach is branch and bound, using a linear relaxation to guide the search. While a uniform discretisation of time is used, which

blows up the number of variables required when planning the day ahead, current MIP solvers can handle thousands of variables so it scales well. By contrast, all performant planning methods use grounding, leading to scaling problems whenever there are more than a handful of objects of any given type, or more than a very small number of parameters in any action schema. The delete relaxation is very general, and its effectiveness depends on the structure of the domain (Hoffmann 2005). If we apply IPC versions of temporal planners to UC instances that MIP finds trivial, we cannot scale beyond managing two or three generators (which is useless considering that there are hundreds of generators in any realistic instance).

However, if we want to consider planning as a practical method to solve problems that extend beyond MIP modelling capabilities, we must first achieve competitiveness on the linear problems. In this paper we address the challenge of making temporal planning competitive with MIP in modelling and solving the traditional day-ahead UC problem. We underline the challenges that standard versions of planners (current IPC versions) face, and explain our approaches to overcoming these challenges.

In the first part of the paper we provide the reader with the formulation of the UC problem and the relevant literature. Then we explain the challenges of this problem from the perspective of AI Planning. The remainder of the paper is dedicated to describing the approaches we have taken to solving the UC as a planning problem, and an evaluation of each approach considered. We present experimental results that show that we have indeed achieved competitive performance on realistic UC instances with linear constraints. The paper concludes with a discussion of the future directions for this work, which concern extending the planning models beyond what can be captured by MIP methods.

2 Formulation of the Problem

We provide here the formalisation of the UC problem.

We consider a set of n units $\mathcal{U} = \{u_1, ..., u_n\}$. Each unit is characterised by the following set of constant parameters:

- $G_{max} \in (0, +\infty)$: maximum stable generation level that the unit cannot exceed (MW);
- $G_{min} \in [0, G_{max})$: minimum stable generation level, as well as power level at the startup and shutdown of the unit (MW);
- $\underline{T}_{on} \in (0, +\infty)$: minimum uptime of the unit (h);
- $\underline{T}_{off} \in (0, +\infty)$: minimum downtime of the unit (h);
- $R_{-/+} \in (0, +\infty)$: maximum ramp down/up rate (MW/h);

In addition, each unit is characterised by a set of variables that change over time:

- $\sigma \in \{on, off\}$: binary variable that indicates if a unit is on or off;
- $\tau \in (0, +\infty)$: time from which the unit has changed status (h);
- $p \in [G_{min}, G_{max}]$: amount of power that a unit is generating (MW);

We consider an interval of time $T = [t_{min}, t_{max})$ and a demand profile $D(t) : T \to [0, +\infty)$ (in MW). The UC is the problem of finding the output of each unit such that the total output is always greater than the demand, while minimising the total cost. Therefore, using the above formulation, the decision variables of the problem are σ_i and p_i.

The cost of a unit is defined as the sum of a startup cost and a production cost. The startup cost $C_{startup}$ depends in general on the time at which the unit is off, however in the remaining of the paper we assume it to be constant. The production cost C_{prod} at time t depends on the power that the unit is generating, and it is often approximated as a quadratic function:

$$C_{prod}(p) = C_{noload} + C_{rate} \cdot p + C_{quad} \cdot p^2, \quad (1)$$

where C_{noload}, C_{rate} and C_{quad} are constant coefficients. The total cost of production of a unit for the period of time T is therefore:

$$\mathcal{C}_{prod} = \int_{t_{min}}^{t_{max}} \left(C_{noload} + C_{rate} \cdot p + C_{quad} \cdot p^2\right) \mathrm{d}t \quad (2)$$

The total cost is the sum of the cost of each unit.

The combinatorial nature of the problem and the lack of availability of a precise forecast of the demand justify the discretisation of the time.

2.1 Related Work

The UC problem in its simplified version with discretised time has been studied since 1960 (Baldwin, Dale, and Dittrich 1959). First popular techniques are based on dynamic programming (Lowery 1966; Le et al. 1983), which cannot scale-up due to memory issues, and Lagrangian relaxation (Muckstadt and Koenig 1977; Merlin and Sandrin 1983; Bertsekas et al. 1983).

Alternative approaches taken from artificial intelligence have recently been developed. Tabu search with embedded priority lists is presented by Mori and Matsuzaki (2001), although the ramp constraint cannot be represented. A coupling between simulated annealing and dynamic economic dispatch is presented by Simopoulos, Kavatza, and Vournas (2006), solving problem instances up to 100 units. Simulated annealing algorithms, however, require fine parametertuning to achieve competitive performance. Genetic Algorithms (Kazarlis, Bakirtzis, and Petridis 1996; Damousis, Bakirtzis, and Dokopoulos 2004) and Particle Swarm Optimisation (Gaing 2003; Pappala, Erlich, and Member 2010) have also been developed to solve the UC.

In recent years the power engineering community has been shifting towards the use of Mixed Integer Programming (MIP) solvers, relying on Branch and Bounds methods. In industry, the MIP formulation has been recently applied by a number of System Operators (SOs) (O'Neill 2011; Hui, Yu, and Moorty 2009). In academic research, tighter or more compact models are studied (Rajan and Takriti 2005; Carrión and Arroyo 2006; Ostrowski, Anjos, and Vannelli 2012; Morales-España, Latorre, and Ramos 2013).

3 Challenges for Planning

The UC problem, as formulated in the previous section, can be seen as a numeric-temporal planning problem, where temporal constraints are given by the minimum uptime and downtime periods of each unit. The problem is characterised by a fixed time horizon. In this, the UC problem is similar to the AC voltage control problem (Piacentini et al. 2015), where AI planning was adopted in distribution networks to maintain the voltage within given boundaries. In both problems there are numeric exogenous events that determine the values of some numeric fluents *(uncontrollable numeric fluents)*, and there is a *mixed metric constraint*. According to the definition presented by Piacentini et al. (2015), this is a comparison of numeric values in which both controllable and uncontrollable numeric fluents appear. *Controllable numeric fluents* are fluents modified only by the effects of actions. In the UC problem the uncontrollable numeric fluents represented the way that consumer demand changes over time, while the requirement that total supply is always greater than the demand, is a mixed metric constraint. Since this constraint need to be satisfied for the entire duration of the plan, the UC problem can be classified as a *bounded trajectory management problem*. However, the UC differs from the AC voltage control problem because we are not only interested in the feasibility of the solution, which can be trivially achieved by switching on all the units, but in getting as close as possible to the quality of MIP solutions.

In contrast to the standard IPC domains, which have uniform action costs, the UC problem is characterised by *state-dependent* action costs. Although it is straightforward to model them in PDDL2.1, since the metric is an expression of numeric fluents, and numeric effects can be defined in terms of complex relations, these kinds of problems have received little attention in the planning community. A recent work by Ivankovic et al. (2014), considering global numerical state constraints, started to deal with state-dependent action costs. The UC problem can be expressed according to the formalism presented in that paper: the status of each unit σ_i is represented by the state variables, or *primary* variables, while the power p_i can be seen as a *secondary variable*, which do not have to have finite domains. The satisfaction of the demand is an invariant constraint C_{inv}. However, their approach for state-dependent action costs showed that in the cases examined, blind search was more effective than their heuristic search, concluding that more work needs to be done to tackle these problems in a domain-independent way.

The use of planning for the generation of electrical power, or to supply a deterministic demand, has already been studied. In the work by Fox, Long, and Magazzeni (2012), a forward search planner is used to determine which battery should be used to satisfy a given load profile, while maximising the expected life-time of the batteries. This problem, however, produces a sequential plan where batteries are connected and disconnected, while the UC problem requires concurrent actions and co-ordination between different generating units. A case where concurrent actions are required is the solar array planning problem (Reddy et al. 2011), where the model-based planning system EUROPA$_2$ is used to determine the orientation of the eight solar arrays on board the International Space Station. Our problem is more complicated in terms of the number of objects present in the model, typically hundreds of units, making the scalability an important issue to overcome.

```
(:durative-action serveDemand
:parameters ()
:duration (>= ?duration 0)
:condition (and
    (at start (startPrecondition))
    (at end (endPrecondition))
    (over all (>= (supply) (demand))))
:effect (and
    (at end (served))))
```

Figure 1: `serveDemand` action in PDDL.

```
(:durative-action switchOn
:parameters (?u - unit)
:duration (>= ?duration (timeSwitchOn ?u))
:condition (and (at start (off ?u))
    (at start (can-on ?u)))
:effect (and (at start (not (off ?u)))
    (at end (can-off ?u)) (at start (can-ramp ?u))
    (at start (on ?u)) (at start (not (can-on ?u)))
    (at start (increase (output ?u) (generationMin ?u)))
    (at start (increase (supply) (generationMin ?u)))
    (at start (increase (totalCost) (costStartup ?u)))))
```

Figure 2: `Switch-on` action in PDDL.

4 Initial PDDL+ Model

In this section we present a PDDL+ model for the UC problem. This is an adaptation of the PDDL2.1 model presented in (Campion et al. 2013), which we summarise here, together with the modification that we have made.

4.1 Model

To enforce the constraint of the satisfaction of the demand throughout the entire period up to the horizon of the problem, an *envelope* action `serveDemand` is modelled. This action has a non-zero duration, which can be determined by the planner. Its application can start as soon as the dummy proposition `startPrecondition` becomes true, and end after the point at which `endPrecondition` becomes true. The management of these propositions is done using timed initial literals to ensure that a `serveDemand` envelope covers the whole period of activity of the plan. The action has an `over all` condition requiring that supply is always greater than demand, and an effect that is specified as a dummy goal condition of the problem, as shown in Figure 1. The actions `SwitchOn` and `SwitchOff` can be modelled capturing the minimum on/off time of the unit being switched, and the starting and ending level constraints (Figure 2). The change in the power produced by a unit can be modelled as a durative action with a continuous linear effect on the output of the unit. In this way the ramp-constraint is implicitly maintained, imposing an adequate change rate.

By contrast with the PDDL2.1 model in (Campion et al. 2013), the generating cost of the power is calculated through

```
(:process calculatingRateCost
:parameters (?u - unit)
:precondition (on ?u)
:effect (increase (totalCost)
    (* #t (+ (costNoLoad ?u)
    (* (output ?u) (rateCost ?u))))))
```

Figure 3: `calculateCost` process in PDDL.

a *process*. In PDDL+, processes are a way to express continuous numeric changes that are not triggered by actions, but by conditions that holds a state. Figure 3 shows the process that calculates the generating cost, which is activated when a unit is switched on. In this model we consider a linear cost of production, with respect to the power generated, and a quadratic function with respect to time. The function `(costNoLoad ?u)` is the PDDL rendering of the constant coefficient C_{noload} in Equation 1.

4.2 Initial Results

We first try to solve the continuous problem modelled with the PDDL+ formulation with off-the-shelf planners. Few planners can support the features of this domain (presence of exogenous numeric events, and continuous non-linear effects). Only UPMurphi (Della Penna et al. 2009) can handle all of the required features, so we adopted it to test the model. UPMurphi is a forward search temporal planner, which exploits model checking algorithms to deal with huge state spaces. It handles continuous processes using the discretise and validate approach: given an initial discretisation, the planner searches for a plan, which is then validated against the original continuous model, using VAL (Howey, Long, and Fox 2004). If the plan is not valid, the process iterates with a finer discretisation. Since UPMurphi cannot support metric minimisation, we adopt, in addition, an iterative approach to lower the total cost. This is added to the domain file, by adding a further `over all` condition to the `serveDemand` action on the total cost. This is not shown in the `serveLoad` action because, if we have a planner that takes into account the metric, we do not need that condition. However, it can be modelled as the invariant `(over all (<= (totalCost) (totalCostLimit))` for some limit that we define. In the following we present the plan produced on a test instance of this problem and some experiments on the scalability of our approach.

Test Case. As a test case we use a problem with 2 units and a demand profile with a fixed time discretisation of 1 hour. In this test case, the demand profile contains 6 time points. We run UPMurphi imposing a memory limit of 1 GB, a minimum time-step of half an hour and we impose that the make-span of the plan cannot exceed 6. We validate the plans generated by UPMurphi with VAL2.8, the automatic validation tool for PDDL (Howey, Long, and Fox 2004). We ran all experiments on 2.8Ghz Intel x86-64 processors, with total RAM of 32GB.

In this test case, only 4 valid plans are generated by UPMurphi, and optimality is not reached. The first 3 plans are generated in about 110 seconds, evaluating 4 million states,

u	G_{min} (MW)	G_{max} (MW)	$T_{off/on}$ (h)	R (MW/h)	$C_{startup}$ ($)	C_{noload} ($/h)	C_{rate} ($/MWh)	C_{quad} ($/MW^{2}h)
1	150	455	8	255	4500	1000	16.19	0.00048
2	150	455	8	255	5000	970	17.26	0.00031
3	20	130	5	50	550	700	16.60	0.00200
4	20	130	5	50	560	680	16.50	0.00211
5	25	162	6	60	900	450	19.70	0.00398
6	20	80	3	60	170	370	22.26	0.00712
7	25	85	3	60	260	480	27.74	0.00079
8	10	55	1	135	30	660	25.92	0.00413

Table 1: Units input data.

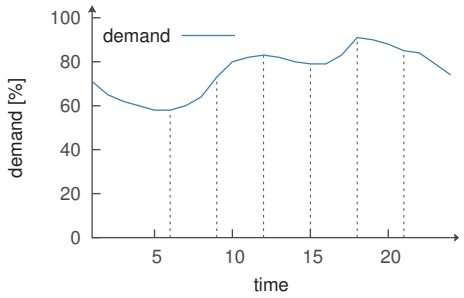

Figure 4: Typical demand profile.

while the last plan is generated in 260 seconds evaluating 9 million states. A fifth plan is generated but the result was invalid, due to the coarse minimum discretisation that we set. Using a finer discretisation UPMurphi runs out of memory and does not produce any plan.

Scalability. Starting from the test case problem, we produce a set problem instances with increasing numbers of units and numbers of time-points. The specification of the generators are taken from the paper (Morales-España, Latorre, and Ramos 2013) and are summarised in Table 1. For these experiments we consider only a linear cost, therefore the quadratic cost C_{quad} is neglected. The demand profile follows a typical distribution, shown in Figure 4.

We generated 49 problems with from 2 to 8 generators and a demand profile composed by 6, 9, 12, 18, 21, and 24 time-points. We solve these problems with UPMurphi, imposing a memory limit of 1 GB. Among all the 49 problems only the two smallest ones are solved: the ones with 2 generators and 6 and 9 time-points. For all the other problems UPMurphi runs out of memory.

The main limitation of this approach is that the standard version of UPMurphi performs an uninformed search algorithm (breadth first search). Moreover, the search spends a lot of time trying to find the exact output of each unit, rather than focusing on the configuration (on/off) of the units.

5 Model with the Decomposition

In order to provide the search with a heuristic guidance and to alleviate the search problem, we recognise that we can decompose the problem into two interleaving sub-problems: a combinatorial search problem, given by the possible configurations of on/off units, and a numeric optimisation problem, the Economic Dispatch (ED) problem, which determines the output of each generator such that the demand is satisfied at the minimum cost. It should be noted that once the

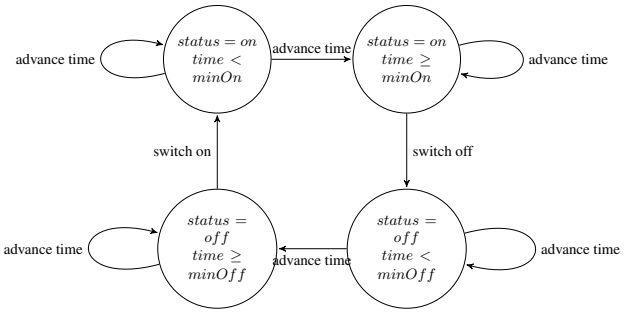

Figure 5: FSM for the status of a unit.

status of each unit is determined, the optimal total cost is unequivocally determined because of the monotonicity of the cost function. With this approach we abandon the continuous model of time, and instead adopt a discretisation of time, similar to a more traditional MIP approach.

5.1 Search Problem

The variables of the search problem represent only the (on/off) status of each generator and the time. Because the numerical optimisation problem refers to an extended period of time, in order to unequivocally determine the numeric optimisation problem, we need to keep track of any changes in the status of the units that happened in the previous time-points. A state in the search space consists of the current time-point, t^*, an array Σ containing the status σ of each unit in each time point preceding t^* and an array T of the status τ of each unit at the current time:

$$s =< t^*, \Sigma, \mathrm{T} > . \tag{3}$$

Following the discretised approach of UPMurphi we need an action that represents the passage of time. This action can be applied only if the numerical optimisation problem associated with the state has a feasible solution, meaning that the demand can be satisfied until the current time-point. The effect of this action is to update the time to a successive time point, consequently changing the demand and the τ_{i,t^*} of each unit. We use a variable discretisation approach, where the passage of time action is parametrised and allows the passage of several time-steps at once. This creates more applicable actions, but it adds the interesting possibility of reaching the end of the time horizon with fewer actions.

An action is needed to switch on a unit at time t^*. This action can be performed on a unit u_i if the unit has been off for a period of time longer that the minimum time period $\underline{T}_{off,i}$. The effect of such an action is to change the status of the unit from off to on and reset the variable τ_i. Analogously the action for switching off a unit can be defined.

The finite state machine representing the change of the status of a unit is represented in Figure 5.

The initial state of a UC problem is determined by the initial σ_{i,t_0} and τ_{i,t_0} of each unit at the first time-point considered, while the goal state is the state where the current time is the last time-point and the configuration of the units is such that the numeric optimisation problem has a feasible solution.

Model 1 Economic Dispatch QP Model.

Inputs:

$\mathcal{U}$: units; T: time-points; $D(T)$: demand profile; t^*: current time.

Variables:

$p_{i,t}$ power generated by unit i at time t

Minimise:

$$\sum_{t<t^*} \sum_{i=1}^{n} \left(C_{rate,i} p_{i,t} + C_{quad,i} p_{i,t}^2 \right) \tag{M1.1}$$

Subject to:

$$\sum_{i=1}^{n} p_{i,t} \geq D_t \qquad \forall t < t^* \tag{M1.2}$$

$$G_{min,i}\sigma_{i,t} \leq p_{i,t} \leq G_{max,i}\sigma_{i,t} \qquad \forall i = 1,...,n;\ \forall t < t^* \tag{M1.3}$$

$$p_{i,t} - p_{i,t-1} \leq -R_i\sigma_{i,t}\sigma_{i,t-1} + P_{min,i}(\sigma_{i,t} - \sigma_{i,t-1}) \qquad \forall i = 1,...,n;\ \forall t < t^* \tag{M1.4}$$

$$p_{i,t} - p_{i,t-1} \geq R_i\sigma_{i,t}\sigma_{i,t-1} + P_{min,i}(\sigma_{i,t} - \sigma_{i,t-1}) \qquad \forall i = 1,...,n;\ \forall t < t^* \tag{M1.5}$$

5.2 Numeric Problem

Each state of the search problem is associated with a numerical optimisation problem, the ED problem, that determines the output of each generator at the minimum cost. Assuming that the cost of production is quadratic, the problem can be formulated as quadratic programming (QP) model (Model 1). It should be noted that in this model the quantities $\sigma_{i,t}$ are not variables, but are input parameters. The objective function (M1.1) accounts only for the rate and the quadratic costs ($C_{rate,i}$ and $C_{quad,i}$ respectively). Constraint (M1.2)) is the demand constraint, which states that the total output of the units must satisfy the total demand. Constraint (M1.3) models the generation limits of units, imposing 0 output for the offline units and an output between G_{min} and G_{max} for the online units. Constraints (M1.4) and (M1.5) capture the ramping limitation and the startup and shutdown capabilities.

5.3 Heuristic and Cost

Each state s that admits a solution of the ED problem is associated with a total cost:

$$c(s) = \sum_{i=1}^{n} \sum_{t=t_0}^{t^*} (C_{noload,i} + C_{rate,i} \cdot p_{i,t}) \cdot \Delta t + \sum_{i=1}^{n} \sum_{t=t_0+\Delta t}^{t^*} C_{startup,i}\sigma_{i,t}(1 - \sigma_{i,t-1}), \tag{4}$$

where Δt is the discretisation step.

It should be noted that using our formulation of the problem, the cost does not depend on the path to reach the state, but it is completely determined by the state. The problem also requires a minimum cost goal state to be found.

In order to estimate the value of a state we solve a relaxed version of the ED model for the time-points $t > t^*$ by replacing the constraints (M1.3-5) with:

$$0 \leq p_{i,t} \leq G_{max,i}\ \forall t \geq t^* \tag{5}$$

and using a linear cost, neglecting the C_{quad} coefficient.

The relaxed problem is solved and the values of $p_{i,t}$ are used to calculate the estimated cost.

$$c(s) = \sum_{t>t^*} (C_{noload,i} + C_{rate,i} \cdot p_{i,t}) \cdot \Delta t \tag{6}$$

A second case arises when the configuration of the units does not provide a feasible solution. In this case we can still associate a cost and a heuristic value to those states, solving the ED problem assuming that also the units that are switched off before time t^* can produce power. For these units we apply the relaxed constraints, while the units that are on in the state that we are evaluating are subjects to the full set of constraints (M1.2-5). If the maximum demand does not exceed the maximum capacity of the system, this relaxed version of the problem has a feasible solution, since the total power generated is subject to a lower bound only. The power of the committed units contributes to the cost of the state, as indicated in Eq. (4), while the power of the uncommitted units contributes to the heuristic value as stated in Eq. (6).

5.4 Implementation Details

For the model presented in this section we use UPMurphi, due to the easiness of implementing a customised heuristic function.

An external procedure is necessary to solve the ED problem associated to each state that gives the cost of a state and the heuristic evaluation of the distance from the state to the goal. This external procedure is implemented as an external function that is called after the application of every action.

The external procedure takes as input the values of t^* and $\sigma_{i,t}$ and τ_{i,t^*} and it builds a QP model to solve the ED. In order to avoid multiple calls to the external procedure we build a unique model for both the ED problem until time t^* and the calculation of the heuristic (Model 2). For this we define a set $\mathcal{X}$ of pairs of indexes (i,t), for which we calculate the ED in a relaxed way (we call them *relaxed units*). At the beginning the $\mathcal{X}$ is initialised with (i,t) such that $i = 1,...n$ and $t > t^*$.

The model is then solved with the CPLEX software and the information on the cost and the heuristic are passed back to the planner. If the solver does not find a solution to the problem, a new model is created updating the set $\mathcal{X}$. The new set is obtained by adding the indices (i,t^*) such that $\sigma_{i,t^*} = 0$. If the solver does not produce a solution, than the heuristic value of the state is assigned to ∞.

The cost of a state and the heuristic are combined in a weighted evaluation function used in the weighted A* algorithm:

$$f(s) = w \cdot c(s) + h(s) \tag{7}$$

The weight w can be specified as input of the planner and it represents a trade off between speed of the planner and quality of the solution.

5.5 Results using the Decomposition Model

In this section we show the results obtained with this approach. We conduct two sets of experiments. The first one is composed by the same problems described in Section 4.2, while in the second set we consider larger problem instances.

Smaller Set. In this set of experiments we take the input data from the previous section, hence we consider a linear cost of production. However, the objective functions in the two approaches are not directly comparable, due to the discretisation adopted with the second approach. We configure UPMurphi to use a weight $w = 1$ and we impose a memory limit of 1 GB and a time limit of 30 minutes. Among the 49 problems, 46 problems are solved, as shown in Table 2.

Model 2 Cost and heuristic calculation.

Inputs:

$\mathcal{U}$: units; T: time-points; $D(T)$: demand profile; $\mathcal{X}$: relaxed units

Variables:

$p_{i,t}$ power generated by unit i at time t

Minimise:

$$\sum_{t\in T}\sum_{i=1}^{n} C_{rate,i}p_{i,t} + \sum_{(i,t)\notin\mathcal{X}} C_{quad,i}p_{i,t}^2 \tag{M2.1}$$

Subject to:

$$\sum_{i=1}^{n} p_{i,t} \geq D_t \qquad \forall t \in T \tag{M2.2}$$

$$G_{min,i}\sigma_{i,t} \leq p_{i,t} \leq G_{max,i}\sigma_{i,t} \qquad \forall (i,t) \notin \mathcal{X} \tag{M2.3}$$

$$0 \leq p_{i,t} \leq G_{max,i} \qquad \forall (i,t) \in \mathcal{X} \tag{M2.4}$$

$$p_{i,t-1} - p_{i,t} \leq -R_i\sigma_{i,t}\sigma_{i,t-1} + P_{min,i}(\sigma_{i,t} - \sigma_{i,t-1}) \qquad \forall (i,t) \notin \mathcal{X} \tag{M2.5}$$

$$p_{i,t-1} - p_{i,t} \geq R_i\sigma_{i,t}\sigma_{i,t-1} + P_{min,i}(\sigma_{i,t} - \sigma_{i,t-1}) \qquad \forall (i,t) \notin \mathcal{X} \tag{M2.6}$$

unit \ time	6	9	12	15	18	21	24
2	0.43	0.44	0.47	0.49	0.53	0.55	0.58
3	0.42	0.46	0.50	0.52	0.57	0.61	0.67
4	0.42	0.48	0.53	0.59	0.67	0.49	0.84
5	0.44	0.52	0.60	0.64	0.79	0.89	1.00
6	0.44	0.54	0.63	0.73	0.89	1.00	1.20
7	0.45	0.75	1.00	1.10	1.40	1.60	1.90
8	0.42	0.54	1.10	1.30	-	-	-

Table 2: Time (sec.) spent to solve each instance of the smaller set using the model with decomposition.

Larger Set. For this set of experiments we generate problems duplicating all the units in Table 1 and we consider 8, 16, 24, 32, 40, 48, 96, 192, and 400 units. The load profile of 24 hours is discretised with a step of half an hour. The cost is assumed to be quadratic.

In order to make the search problem more manageable, we prune the search space in a domain specific way. Since the biggest units (units 1 and 2) are running in the background and are never switched off, we can prune any states in which they are not both running. Furthermore, in real world situations, generating units are infrequently switched on or off. Therefore we impose a constraint that each unit cannot be switched on or off more than 4 times in a day.

We run UPMurphi with a memory limit of 1 GB and a time limit of 2 hours. This reflects the available time to solve the day-ahead unit commitment problem in industry.

For each problem we run UPMurphi with a different weight in the evaluation function. We run the experiments using a fixed and a variable discretisation. In the first case we use a discretisation of 0.5 hour, while in the second case we use time-steps of 0.5, 1, 2 and 4 hours. In Table 3 we report for each problem the total cost and the number of states evaluated varying the weight of the evaluation function. As expected, for each problem, as we increase the weight in the evaluation function, the cost function of the solution decreases, while the number of states evaluated increases.

From this table we can see the impact of variable discretisation. In general, a fixed discretisation produces a higher quality solution, but at the price of a larger number of states evaluated. In some cases the variable discretisation can find solutions for higher weights, finding better quality solutions. In the 192 units problem, the fixed discretisation cannot find any solution. The problem with 400 units is not solved with any weight, with neither variable nor fixed discretisation.

6 Exploiting the Symmetry

In realistic problems, it can be the case that different units have similar characteristics and can be treated as equivalent to one another. We can exploit the symmetry between the units to reduce the number of reachable states. Given two units, u_i and u_j, we define $u_i \equiv u_j$ *(u_i is equivalent or symmetric to u_j)* iff $C_{startup,i} = C_{startup,j} \wedge C_{noload,i} = C_{noload,j} \wedge C_{rate,i} = C_{rate,j} \wedge C_{quad,i} = C_{quad,j} \wedge G_{max,i} = G_{max,j} \wedge G_{min,i} = G_{min,j} \wedge R_{-/+,i} = R_{-/+,j}$.

We can partition the set $\mathcal{U}$ in m subsets $\mathcal{G}_i = \{u_{i_1}, ..., u_{i_o}\}$ such that $\forall j, k \; u_j \equiv u_k$. In other words, the subset $\mathcal{G}_i$ (i^{th} symmetry group) is composed by all equivalent units. This partition is static. At a given time-point, we can partition $\mathcal{G}_i$ into two subsets: $\mathcal{G}_{on,i} = \{u | u \in \mathcal{G}_i \wedge \sigma = on\}$ and $\mathcal{G}_{off,i} = \{u | u \in \mathcal{G}_i \wedge \sigma = off\}$, containing respectively the units that are on and off. Furthermore, we define the sets $\mathcal{G}_{unlock-on,i} = \{u | u \in \mathcal{G}_i \wedge \sigma = on \wedge \tau \geq \underline{T}_{on}\}$ and $\mathcal{G}_{unlock-off,i} = \{u | u \in \mathcal{G}_i \wedge \sigma = off \wedge \tau \geq \underline{T}_{off}\}$, which are subsets of $\mathcal{G}_{on,i}$ and $\mathcal{G}_{off,i}$ respectively. Using this representation, we express the actions in terms of these symmetry groups rather than in terms of units: instead of switching on (or off) a unit, we decrease the cardinality of a $\mathcal{G}_{unlock-on}$ (or $\mathcal{G}_{unlock-off}$) group and increase the cardinality of the respective $\mathcal{G}_{off}$ (or $\mathcal{G}_{on}$). This action automatically selects a unit from these groups to be switched on (or off). In this way the information of every unit is still known and the same ED model can be applied to the state. When the time is passing the τ_i of each unit is updated and if a unit has $\tau_i > \underline{T}_{on}$ (or $\tau_i > \underline{T}_{off}$), than the unit be inserted in the respective $\mathcal{G}_{unlock-on}$ ($\mathcal{G}_{unlock-off}$) group.

Using this representation, the applicable actions from a state are reduced. If we assume that we have m symmetry groups with each group containing l units ($n = m \cdot l$), at a fixed time point, the number of distinct states (without considering any further constraint) is $(l+1)^m$ ($= (l+1)^{\frac{n}{l}} \leq 2^n$ for $n > 1$ and $l \geq 1$).

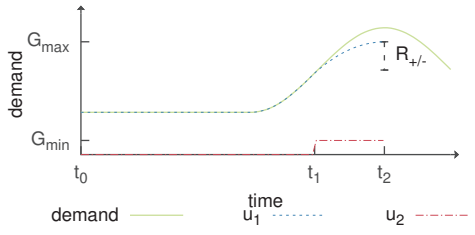

Figure 6: Example of a state not considered in the group representation due to the ramp constraint.

An issue arising with this formulation is that the presence of the ramp constraints (M1.4-5) might introduce a break in the symmetry that we are not considering and could lead to the exclusion of states that should be taken into account. An example of symmetry breaking is shown in Figure 6. In this example the units u_1 and u_2 belong to the same symmetry group. If there is a sharp decrease of the demand (in the figure at time t_2), we cannot indifferently switch off u_1 or u_2, since u_1 must reach the minimum stable generation level G_{min} before it can be switched off, but it is not possible because of the maximum ramp down rate $R_{+/-}$. This case however, should only manifest when there are sharp changes in consumer demand, which does not happen in reality because the demand curve is an average over the entire power system and tends to be smooth.

6.1 Results

To test the model with symmetry we take the larger set of problems described in the previous section and we run UPMurphi with the same memory and time limits, varying the weight of the evaluation function and using the variable discretisation. Each problem contains 8 symmetry groups. The result from this set of experiments is that, in every problem considered, the symmetry model finds exactly the same solutions as found using the model without symmetry, but more efficiently and by exploring fewer states (or the same number of states, such as in case of the 8 units problem, where each group is composed of only one unit). Moreover, with the symmetry model, some problems (16, 24, 32, 40, 48, and 192 units problems) are solved with an higher value of the weight, finding solution with better qualities, and the previously unsolved 400 units problem is solved when symmetry is exploited.

Comparison with MIP. We now compare the results obtained with planning using the decomposed model with symmetry, with the solution obtained with a MIP approach. The MIP model is taken from the paper (Morales-España, Latorre, and Ramos 2013) and adapted to consider the same quadratic objective function and the same constraints. The MIP model is solved using the CPLEX solver, using a memory limit of 4GB and a time limit of 2 hours.

For the planning approach we want also to generate the best solution that can be found before the time limit. Since UPMurphi is not equipped with an anytime search, we emulate the behaviour of the anytime search using an iterative approach in which we impose a total limit on the cost given by the previous solution.

In Table 4 we report the best quality solution for the MIP, and the planning approach using the fixed weight of (0.92) and the best weight for each problem, where the best is selected after trying all the values in our range. We do not report in the table the execution time because in almost all the cases the MIP and the planning approaches consumed all the 2 hours time available. Only the case of 8 and 16 units are solved by MIP optimally in respectively 4 and 840 seconds. Instead, in Table 5 we report for each problem the quality of the first solution obtained with planning (with the fixed weight of 0.92) and the quality obtained by the MIP approach

		Results																	
weight		0.90		0.91		0.92		0.93		0.94		0.95		0.96		0.97		0.98		0.99	
# u	discretisation	Cost	#states	Cost	#states	Cost	#states	Cost	#states	Cost	#states	Cost	#states	Cost	#states	Cost	#states	Cost	#states	Cost	#states
8	variable	616532	55	616532	55	616532	55	616532	55	616532	55	616532	55	616532	73	601976	64	600083	72	600083	71
	fixed	604597	324	597354	329	597354	329	597354	329	605321	350	598873	317	602407	358	600140	373	594681	444	592313	468
		1.00		1.01		1.02		1.03		1.04											
	variable	592511	104	591015	112	584121	188	580459	1260	580459	2444										
	fixed	589258	707	584971	1151	584654	1614	575646	2958	575412	6894										
weight		0.90		0.91		0.92		0.93		0.94		0.95		0.96		0.97		0.98		0.99	
16	variable	1233065	85	1233065	85	1216989	115	1201337	143	1185321	136	1184616	125	1174071	156	1173326	154	1171811	2280	1171314	5061
	fixed	1217362	601	1201710	611	1189058	608	1179263	615	1173713	616	1176049	1460	1183873	7348	1173411	14532	1168580	22681	-	-
24	variable	1835005	136	1817374	167	1816601	132	1800249	136	1783569	172	1766927	193	1755272	227	1758530	17045	-	-	-	-
	fixed	1818122	876	1801718	873	1785434	869	1771781	883	1759021	917	1754167	913	-	-	-	-	-	-	-	-
32	variable	2433978	252	2401821	239	2385458	241	2369232	242	2351850	288	2339457	361	-	-	-	-	-	-	-	-
	fixed	2402615	1198	2385476	1191	2369232	1159	2352595	1178	2340203	1201	-	-	-	-	-	-	-	-	-	-
40	variable	3032957	336	3016749	263	2984038	295	2951543	322	2934163	376	2922525	489	-	-	-	-	-	-	-	-
	fixed	3017592	1486	2985540	1480	2967812	1440	2936400	1491	2922897	1512	2932467	3396	-	-	-	-	-	-	-	-
48	variable	3618338	345	3585604	350	3553103	417	3535346	452	3507084	567	-	-	-	-	-	-	-	-	-	-
	fixed	3601986	1769	3569322	1758	3537210	1786	3509693	1852	-	-	-	-	-	-	-	-	-	-	-	-
96	variable	7202439	955	7137206	953	7071811	1207	7020811	1621	-	-	-	-	-	-	-	-	-	-	-	-
	fixed	7157126	3780	7092176	3750	7041530	3920	-	-	-	-	-	-	-	-	-	-	-	-	-	-
192	variable	-	-	14242719	2599	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	fixed	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 3: UPMurphi solution quality (Cost) and number of states (#states) evaluated, for problems with different numbers of units, using different weights in the evaluation function.

u	MIP	P_{fixed}	P_{best}	$\Delta\%$	$\Delta\%$
8	**571274**	609226	579027	-6.64	-1.36
16	**1140257**	1211736	1159662	-6.27	-1.70
24	1709003	1809162	1751141	-5.86	-2.47
32	2277223	2379332	2335327	-4.48	-2.55
40	2846550	2976661	2917435	-4.57	-2.49
48	3415785	3545537	3501035	-3.80	-2.50
96	6828945	7057785	7009130	-3.35	-2.64
192	13658463	14111294	14006150	-3.36	-2.55
400	28503184	29368707	29368707	-3.04	-3.04

Table 4: Quality obtained with the MIP approach and the planning approach with a fixed and the best weight in the evaluation function. In bold the solution guaranteed to be optimal.

u	MIP	P	$\Delta(\%)$	t_P	t_{MIP}
8	571274	616532	-7.9	1.4	0.33
16	1156742	1216989	-5.2	4.5	0.86
24	1713937	1816601	-6.0	7.8	5.8
32	2279588	2385458	-4.6	17	12
40	2870661	2984038	-3.9	24	23
48	3427797	3553103	-3.7	36	32
96	7398394	7071811	**+4.4**	170	197
192	13666761	14125123	-3.4	960	828
400	30822345	29368707	**+4.7**	7100	7200

Table 5: Quality of the MIP obtained within the time of first planning solution, quality of the first planning solution, marginal difference between qualities, time for the planning to find the first solution, and time for MIP to finds a better quality solution than planning.

within the time in which planning found the first solution. The last column of the table reports the time in which the MIP approach finds the first better solution with respect to planning. This data is of interest in the context of the future energy scenario, where the power system is expected to be more dynamic than is the case currently, and we expect to have to re-solve the UC problem much more frequently so that speed of solution becomes more important. From the table we can see that in two cases (96 and 400 units) the planning approach finds a first better solution than MIP. This is not a conclusive evidence for favouring planning over MIP, but it points out the potential benefit of planning.

7 Conclusion

In this paper we have presented the work we have done towards demonstrating that, although domain-independent planners have limitations, with careful domain modelling and some domain specific guidance in the search temporal planning can solve realistic Unit Commitment problems. We started from a full PDDL+ model, for which only one domain-independent planner, UPMurphi, was cable of handling all the characteristic of the domain. We have shown that the planner does not scale when dealing with numerical optimisation problems and without proper heuristic guidance. We then moved to a model where the problem is decomposed into two sub-problems, the pure search problem and the Economic Dispatch which is exploited by the search to determine the cost and heuristic value of a state. We moved from being able to solve only very small instances of the problem (fewer than 10 units) to being able to scale up to 192 units, with solutions of quality similar to those produced by MIP. To further alleviate the search burden, we applied a symmetry model, where units with the same characteristics are grouped together, and the actions are applied to the groups, rather than to single units. With this method we were able to scale up to problems with 400 units. In terms of the quality of the planned solutions, a direct comparison with a traditional MIP approach is made. This comparison shows that although planning is still not competitive with MIP, it can produce solution within $\sim$3% of the ones obtained with MIP. Scrutiny of the solutions shows that the units chosen by the planner and by the MIP solver are almost always the same, and almost always over the same time periods. The variations might give some interesting insights into where the planner makes poor decisions from which it cannot recover, leading to the observed gap in quality. This investigation is a topic for future work.

7.1 Future Work

In this work, we have compared UPMurphi against the MIP approach, using exactly the same linear constraints and

quadratic objective function. Our goal in this work has been to establish a competitive baseline from which to further develop the planning approach. We now want to direct our planning efforts to solving more complex problems in generating the future electrical power supply. For example, storage will significantly affect the Unit Commitment problem, but the problem of energy storage is one that cannot be tackled using MIP, because the non-linear constraints that arise cannot even be modelled. Another problem of great economic importance concerns reducing the spinning reserve while retaining robust solutions. The current approach, which can be modelled in MIP, is to simply over-supply by a fixed (conservative) amount. This is expensive and unnecessary. It would be more interesting to define a buffer of varying width, but this leads to a non-linear constraint. In our future work we will work with our collaborators in the power engineering community to address these considerations.

Acknowledgments

This research was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) as part of the project entitled *The Autonomic Power System* (Grant Ref: EP/I031650/1).

References

Bajada, J.; Fox, M.; and Long, D. 2015. Temporal Planning with Semantic Attachment of Non-Linear Monotonic Continuous Behaviours. In *IJCAI*, 1523–1529.

Baldwin, C. J.; Dale, K. M.; and Dittrich, R. F. 1959. A Study of the Economic Shutdown of Generating Units in Daily Dispatch. *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems* 78(4):1272–1282.

Bertsekas, D. P.; Lauer, G. S.; Sandell, N. R.; and Posbergh, T. A. 1983. Optimal Short-Term Scheduling of Large-Scale Power Systems. *IEEE Transactions on Automatic Control* AC-28(1):1–11.

Campion, J.; Dent, C.; Fox, M.; Long, D.; and Magazzeni, D. 2013. Challenge: Modelling Unit Commitment as a Planning Problem. In *ICAPS*, 452–456.

Carrión, M., and Arroyo, J. M. 2006. A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems* 21(3):1371–1378.

Cashmore, M.; Fox, M.; Long, D.; and Magazzeni, D. 2016. A Compilation of the Full PDDL+ Language into SMT. In *ICAPS*.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. *ICAPS* 42–49.

Damousis, I. G.; Bakirtzis, A. G.; and Dokopoulos, P. S. 2004. A Solution to the Unit-commitment Problem Using integer-coded Genetic Algorithm. *IEEE Transactions on Power Systems* 19(2):1165–1172.

Della Penna, G.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. UPMurphi: A Tool for Universal Planning on PDDL+ Problems. In *ICAPS*, 106–113.

Fox, M., and Long, D. 2003. PDDL2. 1: An Extension to PDDL for Expressing Temporal Planning Domains. *JAIR* 20:61–124.

Fox, M., and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *JAIR* 27:235–297.

Fox, M.; Long, D.; and Magazzeni, D. 2012. Plan-based Policies for Efficient Multiple Battery Load Management. *JAIR* 44:335–382.

Gaing, Z.-L. 2003. Discrete Particle Swarm Optimization Algorithm for Unit Commitment. *IEEE Power Engineering Society General Meeting 2003* 1:418–424.

Hoffmann, J., and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *JAIR* 14(1):253—-302.

Hoffmann, J. 2005. Where "Ignoring Delete Lists" Works: Local Search Topology in Planning Benchmarks. *JAIR* 24:685–758.

Howey, R.; Long, D.; and Fox, M. 2004. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence*, 294–301.

Hui, H.; Yu, C.-N.; and Moorty, S. 2009. Reliability Unit Commitment in the New ERCOT Nodal Electricity Market. In *IEEE Power and Energy Society General Meeting, PES '09*, 1–8.

Ivankovic, F.; Haslum, P.; Shivashankar, V.; and Nau, D. S. 2014. Optimal Planning with Global Numerical State Constraints. In *ICAPS*, 145–153.

Kazarlis, S.; Bakirtzis, A.; and Petridis, V. 1996. A Genetic Algorithm Solution to the Unit Commitment Problem. *IEEE Transactions on Power Systems* 11(1):83–92.

Le, K. D.; Day, J. T.; Cooper, B. L.; and Gibbons, E. W. 1983. A Global Optimization Method for Scheduling Thermal Generation, Hydro Generation, and Economy Purchases. *IEEE Transactions on Power Apparatus and Systems* PAS-102(7):1986–1993.

Lowery, P. G. 1966. Generating Unit Commitment by Dynamic Programming. *IEEE Transactions on Power Apparatus and Systems* PAS-85(5):422–426.

Merlin, A., and Sandrin, P. 1983. A New Method for Unit Commitment at Electricite De France. *IEEE Transactions on Power Apparatus and Systems* PAS-102(5):1218–1225.

Morales-España, G.; Latorre, J. M.; and Ramos, A. 2013. Tight and Compact MILP Formulation for the Thermal Unit Commitment Problem. *IEEE Transactions on Power Systems* 28(4):4897–4908.

Mori, H., and Matsuzaki, O. 2001. Embedding the Priority List into Tabu Search for Unit Commitment. In *Power Engineering Society Winter Meeting*, volume 3, 1067–1072.

Muckstadt, J. A., and Koenig, S. A. 1977. An Application of Lagrangian Relaxation to Scheduling in Power-Generation Systems. *Operations Research* 25(3):387–403.

O'Neill, R. P. 2011. Recent ISO Software Enhancements and Future Software and Modeling Plans. Technical report, Office of Energy Policy and Innovation, Washington D.C.

Ostrowski, J.; Anjos, M. F.; and Vannelli, A. 2012. Formulations for the Unit Commitment Problem. *IEEE Transactions on Power Systems* 27(1):39–46.

Pappala, V. S.; Erlich, I.; and Member, S. 2010. A Variable-Dimension Optimization Approach to Unit Commitment Problem. *IEEE Transactions on Power Systems* 25(3):1696–1704.

Piacentini, C.; Alimisis, V.; Fox, M.; and Long, D. 2015. An Extension of Metric Temporal Planning with Application to AC Voltege Control. *Artificial intelligence* 229:210–245.

Rajan, D., and Takriti, S. 2005. Minimum Up/Down Polytopes of the Unit Commitment Problem with Start-Up Costs. Technical report, IBM Reasearch Division, Yorktown Heights.

Reddy, S. Y.; Frank, J. D.; Iatauro, M. J.; Boyce, M. E.; Kürklü, E.; Ai-Chang, M.; and Jónsson, A. K. 2011. Planning solar array operations on the international space station. *ACM Transactions on Intelligent Systems and Technology* 2(4):1–24.

Simopoulos, D.; Kavatza, S.; and Vournas, C. 2006. Unit Commitment by an Enhanced Simulated Annealing Algorithm. *IEEE Transactions on Power Systems* 21(1):68–76.

Wood, A.; Wollenberg, B.; and Sheblé, G. 2013. *Power Generation, Operation and Control*. New Jersey: Wiley, third edition.