A unit decommitment method in power system scheduling

Chung-Li Tseng and Shmuel S Oren
Department of Industrial Engineering and Operations Research, University of California at Berkeley, Berkeley CA 94720, USA

Alva J Svoboda and Raymond B Johnson
Pacific Gas & Electric Company, San Francisco CA 94177, USA

This paper presents a unit decommitment method for power system scheduling. Given a feasible unit commitment, our algorithm determines an optimal strategy for decommitting overcommitted units based on dynamic programming. This method is being developed as a possible post-processing tool to improve the solution quality of the existing unit commitment algorithm used at PG&E. It can also be integrated into any other unit commitment method or used as a complete unit commitment algorithm in itself. The decommitment method can also be used as a tool to measure the solution quality of unit commitment algorithms. The proposed method maintains solution feasibility at all iterations. In this paper we prove that the number of iterations required by the method to terminate is bounded by the number of units. Numerical tests indicate that this decommitment method is computationally efficient and can improve scheduling significantly. © 1997 Elsevier Science Ltd.

Keywords: power system scheduling, unit commitment, unit decommitment

1. Introduction
The unit commitment problem at a power utility like PG&E requires economically scheduling generating units over a planning horizon so as to meet forecast demand and system operating constraints. It has been an active research subject due to the potential cost savings and the difficulty of the problem. The unit commitment problem is a mixed integer programming problem, and has been proved to be in the class of the most intractable problems, referred to as NP-hard [1]. Many optimization methods have been proposed to solve the unit commitment problem (e.g. [2]). These methods include priority list methods [3], dynamic programming methods [4–6] and Lagrangian relaxation methods [2,7–9]. Lagrangian relaxation methods are now among the most widely used approaches for solving such problems. At PG&E, the Hydro-Thermal Optimization (HTO) program was developed almost a decade ago, based on the Lagrangian relaxation approach [8]. In recent work, the Lagrangian relaxation-based algorithm has been extended to schedule thermal units under ramp constraints [10].

The Lagrangian relaxation approach is a dual method. The basic idea is to relax the system demand and spinning reserve constraints using Lagrange multipliers. The resulting dual problem is then decomposed into unit subproblems, each of which can be easily solved. Despite efficiency in solving the dual problem due to its separability, the solution of the dual problem does not necessarily yield a feasible solution of the original problem. Therefore the Lagrangian relaxation methods for solving unit commitment have often been algorithms with two phases: a dual optimization phase and a feasibility phase (see [1,11,12] for discussion and interpretation). A common phenomenon observed in the feasibility phase is overcommitment of generating units. For the purpose of improving cost savings, our attention has focused on how to improve a feasible unit commitment while maintaining feasibility, by developing decommitment algorithms.

In [13] a new unit commitment method was proposed. The method in [13] resembles the Lagrangian relaxation approach but only the system demand constraints are relaxed with multipliers. However, the multipliers are not updated by a subgradient rule but from an economic dispatch solution. This method starts with a unit commitment with all available units on-line at all hours in the planning horizon, and improves the commitment by gradually decommitting units.

In this paper, we propose a general decommitment algorithm for power system scheduling. Given any feasible unit commitment, the proposed algorithm determines an optimal strategy for decommitting overcommitted units. Without incorporating ramp constraints, the problem is formulated as an integer programming problem and is solved by dynamic programming. The method maintains
solution feasibility at all iterations. We shall show in this paper that the number of iterations required for our method to terminate is bounded by the number of units. The computation involved is fast. Because the problem feasibility is always satisfied, the method provides useful information at any iteration and can serve as an efficient post-processing method for any existing unit commitment algorithm. In this paper, we do not directly address the intractability of the unit commitment problem, but we will discuss applying the proposed decoupling method as a complete unit commitment algorithm.

This paper is organized as follows. In Section II the unit commitment problem is formulated. The unit commitment algorithm and its convergence properties are presented in Section III. In Section IV approximate versions of the decoupling algorithm are discussed to improve algorithm performance. An attempt to apply the unit commitment method presented in Section III as a complete unit commitment method is discussed in Section V. We provide some numerical test results and the conclusions of this paper in Sections VI and VII.

II. Problem formulation

In this paper the following standard notation will be used. Additional symbols will be introduced when necessary.

- \( i \) index for the number of units \((i = 1, \ldots, I)\)
- \( t \) index for time \((t = 0, \ldots, T)\)
- \( u_{it} \) zero-one decision variable indicating whether unit \( i \) is up or down in time period \( t \)
- \( x_{it} \) state variable indicating the length of time that unit \( i \) has been up or down in time period \( t \)
- \( t_{on}^{i}(t_{off}^{i}) \) the minimum number of periods unit \( i \) must remain on (off) after it has been turned on (off)
- \( p_{it} \) state variable indicating the amount of power unit \( i \) is generating in time period \( t \)
- \( p_{i}^{\text{min}}(p_{i}^{\text{max}}) \) minimum (maximum) rated capacity of unit \( i \)
- \( C_{i}(p_{it}) \) fuel cost for operating unit \( i \) at output level \( p_{it} \) in time period \( t \)
- \( S_{i}(x_{i,t-1}, x_{i}, u_{i}, u_{i,t-1}) \) startup cost associated with turning on unit \( i \) at the beginning of time period \( t \)
- \( D_{t} \) demand in time period \( t \)
- \( R_{i} \) spinning capacity requirement in time period \( t \)

The unit commitment problem is formulated as per the following mixed-integer programming problem (note that the underlined variables are vectors in this paper, e.g. \( \mathbf{u} = (u_{11}, \ldots, u_{IT}) \)):

\[
\min_{\mathbf{u}, \mathbf{x}, \mathbf{D}} \sum_{t=1}^{T} \sum_{i=1}^{I} \left[ C_{i}(p_{it})u_{it} + S_{i}(x_{i,t-1}, u_{i}, u_{i,t-1}) \right]
\]

subject to the demand constraints,

\[
\sum_{i=1}^{I} p_{it}u_{it} = D_{t} \quad t = 1, \ldots, T
\]

and the spinning capacity constraints,

\[
\sum_{i=1}^{I} p_{it}^{	ext{max}} u_{it} \geq R_{i} \quad t = 1, \ldots, T
\]

There are other unit constraints such as unit capacity constraints,

\[
p_{i}^{\text{min}} \leq p_{it} \leq p_{i}^{\text{max}} \quad i = 1, \ldots, I; \quad t = 1, \ldots, T
\]

the state transition equation for \( i = 1, \ldots, I \),

\[
x_{it} = \begin{cases} 
\max (x_{i,t-1}, 0) + 1, & \text{if } u_{it} = 1 \\
\min (x_{i,t-1}, 0) - 1, & \text{if } u_{it} = 0
\end{cases}
\]

the minimum up/down time constraints for \( i = 1, \ldots, I \),

\[
\begin{cases} 
u_{it} = 1, & \text{if } 1 \leq x_{i,t-1} < t_{on}^{i} \\
u_{it} = 0, & \text{if } -t_{off}^{i} > x_{i,t-1} \leq 0 \\
0 \text{ or } 1, & \text{otherwise}
\end{cases}
\]

and the initial conditions on \( x_{it} \) at \( t = 0 \) for \( \forall i \).

II.1 Model of cost function

The generating cost of a thermal unit includes fuel costs and the startup costs. In this paper, the fuel cost is modeled as a convex quadratic function of the power output (MWh) of the unit:

\[
C_{i}(p_{it}) = a_{i0} + q_{i1} p_{it} + a_{i2} p_{it}^2 \quad i = 1, \ldots, I
\]

where each of the \( a_{ij} \) coefficients are taken to be nonnegative [14].

The startup costs vary with the temperature of the boiler and therefore depend on the length of time that the unit has been off. They are modeled as an exponential function of time since shutdown. The longer a unit is off, the greater the cost will be to start up the unit. To further simplify the notation, we let \( S_{i}(x_{i}, u_{i}) = S_{i}(x_{i,t-1}, u_{i}, u_{i,t-1}) \).

II.2 Economic dispatch

Economic dispatch is a problem of allocating the system demand among all the on-line generating units at any time in the planning horizon. In this paper, the theory is developed with respect to a given commitment \( \mathbf{u} \) satisfying equations (3), (5) and (6). All variables hatted with a tilde are relaxed to this commitment. We define the index set of on-line units at time \( t \) with respect to this feasible commitment \( J(t; \mathbf{u}) = \{ i | \tilde{u}_{it} = 1 \} \). For simplicity, \( \tilde{J} = J(t; \mathbf{u}) \). The economic dispatch problem is to determine the generation levels of on-line units so as to minimize fuel costs, subject to equations (2) and (4).

\[
\text{edp}(\tilde{J}, t) = \min_{p_{i}^{\text{min}} \leq p_{it} \leq p_{i}^{\text{max}}} \left\{ \sum_{i \in \tilde{J}} C_{i}(p_{it}) \right\} \quad \forall t
\]

\[
\tilde{p}_{i} = \arg \min_{p_{i}^{\text{min}} \leq p_{it} \leq p_{i}^{\text{max}}} \left\{ \sum_{i \in \tilde{J}} C_{i}(p_{it}) \right\} \quad \forall t
\]

where \( \text{edp}(\tilde{J}, t) \) is the economic dispatch value at hour \( t \), and \( \tilde{p} \) is the economic dispatch solution. Note that the economic dispatch is a quadratic programming problem. Its necessary condition of optimality is generally stated as the operation of all generators at equal marginal (incremental) cost (e.g. [6]).
More precisely, there exist $\tilde{\lambda}_i$, $t = 1, \ldots, T$, such that
\[
C_i(\tilde{p}_i) = \tilde{\lambda}_i, \quad \text{for} \ p_i^{\text{min}} < \tilde{p}_i < p_i^{\text{max}}
\]
\[
C_i(\tilde{p}_i) = \tilde{\lambda}_i, \quad \text{for} \ \tilde{p}_i = p_i^{\text{min}}
\]
\[
C_i(\tilde{p}_i) = \tilde{\lambda}_i, \quad \text{for} \ \tilde{p}_i = p_i^{\text{max}}
\]
(10)

for $i = 1, \ldots, I$.

Note that, to guarantee the existence of a solution of edp($J_i, t$) in equation (8), the following capacity requirement is implicitly assumed.
\[
\sum_{i \in J_t} p_i^{\text{min}} \leq D_t \leq \sum_{i \in J_t} p_i^{\text{max}}, \quad \forall t
\]
(11)

III. Unit decommitment method

Given a feasible schedule $(\tilde{\mathbf{u}}, \tilde{\mathbf{p}})$ (satisfying equations (2)–(6) and (10)), we consider the problem of decommitting unit $j$ in hour $t$. Before the decommitment, the total fuel cost in hour $t$ is edp($J_t, t$). Decommitting unit $j$ in hour $t$, its generated power $\tilde{p}_j$ will be distributed to other on-line units in hour $t$ in order to satisfy equation (2). The total fuel generating cost after decommitment is edp($J_t \setminus \{j\}, t$). The increased fuel cost of all on-line units other than unit $j$ due to its decommitment in hour $t$ is denoted by $\Delta C_j(\tilde{\mathbf{u}}, \tilde{\mathbf{p}}, t)$. The exact value of $\Delta C_j(\tilde{\mathbf{u}}, \tilde{\mathbf{p}}, t)$ is
\[
\Delta C_j(\tilde{\mathbf{u}}, \tilde{\mathbf{p}}, t) = \text{edp}(J_t \setminus \{j\}, t) - (\text{edp}(J_t, t) - C_j(\tilde{p}_j, \tilde{\mu}_j))
\]
(12)

Now we consider the following problem $(P_j)$ to improve the commitment of unit $j$ (with all other units' commitments fixed.)

\[
\begin{align*}
(P_j) \quad & \min_{u_j \in \{0, 1\}} \sum_{i=1}^T [(C_i(\tilde{p}_i) + S_j(\mathbf{u}, t)) u_j - \\
& \quad + \Delta C_j(\tilde{\mathbf{u}}, \tilde{\mathbf{p}}, t)(1 - u_j)]
\end{align*}
\]

subject to
\[
u_j = \begin{cases} 0 & \text{if } \tilde{\mu}_j = 0, \\
1 & \text{if } \tilde{\mu}_j = 1 \text{ and } \sum_{i \in J_t \setminus \{j\}} p_i^{\text{max}} < R_t
\end{cases}
\]
and the minimum up/down time constraints in equations (5) and (6) for $i = j$ with initial conditions at $t = 0$. In the sequel, the solution of $(P_j)$ will be called the tentative commitment of unit $j$.

In solving $(P_j)$ with respect to $(\tilde{\mathbf{u}}, \tilde{\mathbf{p}})$ it can be seen that unit $j$ remains off in off-line hours; and may be turned off if its removal will not violate the system spinning capacity requirement. The objective function shows that in an on-line hour, if unit $j$ remains on, the generating cost in that hour is the original fuel cost; while if unit $j$ is turned off, the generating cost in that hour will be $\Delta C_j(\tilde{\mathbf{u}}, \tilde{\mathbf{p}}, t)$. In both cases, the startup cost is imposed when unit $j$ is started up.

Obviously, the tentative commitment obtained in $(P_j)$ with respect to $(\tilde{\mathbf{u}}, \tilde{\mathbf{p}})$ is no worse than \{\tilde{u}_j\}_{j=1}^J$ $\Leftrightarrow$ because \{\tilde{u}_j\}_{j=1}^J$ itself is feasible to $(P_j)$. Note that $(P_j)$ is an integer programming problem and can be solved using either forward or backward dynamic programming. Figure 1 shows the state transition diagram of dynamic programming. Equation (13) can be regarded as a unit availability constraint, and is modeled in the transition cost in the state transition diagram.

Based on a given unit commitment $(\tilde{\mathbf{u}}, \tilde{\mathbf{p}})$, problem $(P_j)$

Figure 1. The state transition diagram
improves the commitment of unit $j$ with the commitments of all other units fixed. If we include the total generating cost of all the units other than $j$ into the objective of $(P_1)$, we have

$$
\min_{u_{ij} \in \{0,1\}} \sum_{t=1}^{T} [(C_j(\bar{p}_j) + S_j(\bar{u}_t, t))u_{jt} + \Delta C_j(\bar{u}_t, \bar{p}_t, t)(1 - u_{jt})] + \sum_{i \neq j} \sum_{t=1}^{T} [C_i(\bar{p}_i)u_{it} + S_i(\bar{u}_t, t)]
$$

(14)

$$
= \min_{u_{ij} \in \{0,1\}} \sum_{t=1}^{T} [(C_j(\bar{p}_j) + S_j(\bar{u}_t, t))u_{jt} + \Delta C_j(\bar{u}_t, \bar{p}_t, t)(1 - u_{jt})] + \sum_{i \neq j} \sum_{t=1}^{T} [C_i(\bar{p}_i)u_{it} + S_i(\bar{u}_t, t)]
$$

(15)

$$
= \min_{u_{ij} \in \{0,1\}} \sum_{t=1}^{T} [(\text{edp}(\bar{J}_j, t) + S_j(\bar{u}_t, t))u_{jt} + \text{edp}(\bar{J}_j, t)(1 - u_{jt})] + \sum_{i \neq j} \sum_{t=1}^{T} S_i(\bar{u}_t, t)
$$

(16)

(Note: if $j \notin J$, let $\bar{J}_j(j) = \bar{J}_j$. Since the last term in equation (14) (also the last term in equation (16)) is a constant, the inclusion of this term in $(P_1)$ will not affect the minimization solution of $(P_1)$. It can be seen from equation (16) that solving $(P_1)$ with respect to $(\bar{u}; \bar{p})$ will determine the optimal strategy to improve $(\bar{u}; \bar{p})$ by only decommitting unit $j$ at some hours.

In the following algorithm, superscript $k$ denotes the $k$th iteration of the algorithm. Let $\bar{L}_i^0$, $i = 1, \ldots, I$ be the total generating cost (fuel cost and startup cost) of unit $i$ of the feasible schedule $(\bar{u}_t^0; \bar{p}_t^0)$; and $\bar{L}_i^k$, $i = 1, \ldots, I$, be the optimal objective value of $(P_k^k)$ solved with respect to feasible solution $(\bar{u}_t^k; \bar{p}_t^k)$. We now state the decommitment algorithm:

**Generic unit decommitment algorithm**

Data: Feasible solution $(\bar{u}_t^0, \bar{p}_t^0)$ and the corresponding $\bar{L}_i^0$, $i = 1, \ldots, I$ are given.

Step 0: $k = 0$.

Step 1: Solve $(P_k^1)$ with respect to $(\bar{u}_t^k; \bar{p}_t^k)$ and obtain $\bar{L}_i^k$ for all $i = 1, \ldots, I$.

Step 2: Select a unit $m$ such that $(\bar{L}_m^k - \bar{L}_m^0) > 0$. If there is no such unit, stop; otherwise update the commitment of unit $m$ in $(\bar{u}_t^k)$ by the commitment obtained in $(P_k^1)$. The resultant unit commitment is assigned to be $\bar{u}_t^{k+1}$.

Step 3: Perform the economic dispatch on $(\bar{u}_t^{k+1})$ to obtain $\bar{p}_t^{k+1}$ and evaluate $\bar{L}_i^{k+1}$, the total generating cost of unit $i$, $i = 1, \ldots, I$.

Step 4: $k = k + 1$, go to Step 1.

At each iteration, the tentative commitment problem $(P_t)$ of each unit $i$ is obtained, and the potential savings for running the tentative commitment $\bar{L}_i - \bar{L}_i$ are also calculated. The algorithm chooses the tentative commitment which can yield savings to replace the original commitment. The rule for selecting a unit in Step 2, corresponding to choosing a descent direction as in continuous optimization theory, is not unique. For example,

- $m = \text{arg max} \{(\bar{L}_i - \bar{L}_i^0) | i = 1, \ldots, I\}$ — the steepest descent direction (e.g. [15]).
- $m = \text{arg max} \{((\bar{L}_i - \bar{L}_i^0)/\theta) | i = 1, \ldots, I\}$ — the deflected gradient direction (e.g. [15]), where $\theta$ can be, say, $\bar{p}_t^{\max}$. The rule used in [13] belongs to this category.
- Assuming $C_i(p) = a_{0i} + a_{1i}p + a_{2i}p^2$ and $a_{2i} \geq a_{2j} \cdots \geq a_{2r}$, $m$ is the smallest index $i$ that has not been selected in previous iterations such that $\bar{L}_i - \bar{L}_i^0 > 0$ — the coordinate descent method (e.g. [9]). This process is repeated if more than one cycle is needed.

Theoretically, different unit selection rules in Step 2 may yield different convergences. However, our experience shows that the performance of the algorithm is insensitive, at least to the above three selection rules. Issues about the evaluation of $\Delta C_j(\bar{u}_t, \bar{p}_t, t)$ will be discussed in Section IV.

### III.1 Convergence analysis

In this section, we discuss the convergence properties of the decommitment algorithm. We will show the conditions under which the algorithm will terminate within a finite number of iterations, with the number of iterations bounded by the number of units.

**Lemma 1**

Given a feasible solution $(\bar{u}; \bar{p})$ and its associated $\lambda$, for unit $j$, $\Delta C_j(\bar{u}; \bar{p}, t) \geq \lambda \Delta \bar{p}_t$, for all $t$. \hfill $\blacksquare$

**Proof:** Assume that solving $\text{edp}(\bar{J}_j(j), t)$ yields generation level $\bar{p}_t + \Delta \bar{p}_t$ for unit $i \in \bar{J}_j(j)$ such that $\sum_{i \in \bar{J}_j(j)} \Delta \bar{p}_t = \Delta \bar{p}_t^k$. Since all the fuel cost functions are assumed to be smooth, from equation (7), we have

$$
\Delta C_j(\bar{u}_t, \bar{p}_t, t) = \sum_{i \in \bar{J}_j(j)} C_i(\bar{p}_t + \Delta \bar{p}_t) - C_i(\bar{p}_t) \geq \sum_{i \in \bar{J}_j(j)} C_i(\bar{p}_t) \Delta \bar{p}_t ^k \\
\geq \lambda \sum_{i \in \bar{J}_j(j)} \Delta \bar{p}_t ^k = \lambda \Delta \bar{p}_t
$$

(17)

(Note that for those $i$ such that $C_i(\bar{p}_t) \leq \lambda$, $\Delta \bar{p}_t = 0$.) \hfill $\blacksquare$

**Lemma 2**

Given a feasible solution $(\bar{u}; \bar{p})$ and its associated $\lambda$, for unit $j$ in time $t$ the following statements are true.

(i) If $\bar{u}_t = 0$, $\Delta C_j(\bar{u}_t, \bar{p}_t, t) = 0$.

(ii) If $\bar{u}_t = 1$, $\Delta C_j(\bar{u}_t, \bar{p}_t, t) > 0$.

(iii) If $\bar{u}_t = 0$ at the $k$th iteration, i.e., $\bar{u}_t^k = 0$, then $\bar{u}_t^{k+1} = 0$ and $\Delta C_j(\bar{u}_t^{k+1}, \bar{p}_t, t) = 0$ for all $t$. \hfill $\blacksquare$

(iv) Before unit $j$ is decommitted in time $t$ at some iteration, \{$(\bar{p}_t^k)$, $(\bar{p}_t^{k+1})$ and $(\bar{L}_i^k)$\} are nondecreasing sequences in $k$, for all $t$.

(v) Before unit $j$ is decommitted in time $t$ at some iteration, \{(\bar{L}_i^k - C_j(\bar{p}_t^k))\} is a nondecreasing sequence in $k$.

(vi) $(\Delta C_j(\bar{u}_t^{k+1}, \bar{p}_t^{k+1}, t))$ is a nondecreasing sequence in $k$ before unit $j$ is decommitted in time $t$ at some iteration. \hfill $\blacksquare$

**Proof:** Statements (i)–(iii) are obvious. Since the algorithm only involves unit decommitment and the load balance equation is satisfied at all iterations, (iv) is true. To
prove statement (v), note that
\[ C_j(\tilde{p}_j^k) - \tilde{X}_k^k \hat{p}_j^k = C_j(p_j^{\min}) + \int_{p_j^{\min}}^{p_j^k} \left( \frac{C_j(p) - \tilde{X}_k^k}{p - \tilde{X}_k^k} \right) dp - \tilde{X}_k^k \hat{p}_j^k \]
\[ \tag{18} \]

It can be shown that the second term on the right hand side of equation (18) (with an integral) is nonincreasing in k, since for any iteration k such that \( \tilde{X}_k^k \leq C_j(p_j^{\min}) \) this term remains zero, and if \( C_j(p_j^{\min}) < \tilde{X}_k^k \), then \( \frac{C_j(p) - \tilde{X}_k^k}{p - \tilde{X}_k^k} \leq 0 \), for \( p \in (p_j^{\min}, \tilde{X}_k^k) \). The last term in equation (18), \( -\tilde{X}_k^k \hat{p}_j^k \) is nonincreasing in k from (iv). So \( \{\tilde{X}_k^k \hat{p}_j^k - C_j(\tilde{p}_j^k)\} \) is a nondecreasing sequence in k.

To prove statement (vi), assume at iteration k, the generation levels of an on-line unit \( i \neq j \) will increase \( \Delta p_i \) due to the decommitment of unit j. We have
\[ \Delta C_j(\tilde{u}_j^k, \tilde{p}_j^k, t) = \min_{\Delta p_i} \sum_{i \in J_k(\tilde{t})} C_i(\tilde{p}_i^k + \Delta p_i) - C_i(\tilde{p}_i^k) \]
\[ \text{s.t.} \quad \sum_{i \in J_k(\tilde{t})} \Delta p_i = \tilde{p}_j^k \]
\[ 0 \leq \Delta p_i \leq \tilde{p}_i^{\max} - \tilde{p}_i^k, \quad \forall i \in J_k(\tilde{t}) \]
\[ \tag{19} \]

Consider two iterations \( k' \) and \( k'' \) such that \( k'' < k' \) and \( \Delta C_j(\tilde{u}_j^{k'}, \tilde{p}_j^{k'}, t), \Delta C_j(\tilde{u}_j^{k''}, \tilde{p}_j^{k''}, t), \Delta C_j(\tilde{p}_j^{k'}, \tilde{p}_j^{k''}, t) \) are nonzero. Note that \( J_k^{k'} \subseteq J_k^{k''} \), \( \tilde{p}_j^{k'} \leq \tilde{p}_j^{k''} \) and \( 0 < C_i(\tilde{p}_i^{k'}) \leq C_i(\tilde{p}_i^{k''}) \) for all \( i \in J_k^{k'} \). Based on this information, it is straightforward to show that \( \Delta C_j(\tilde{u}_j^{k'}, \tilde{p}_j^{k'}, t) \leq \Delta C_j(\tilde{u}_j^{k''}, \tilde{p}_j^{k''}, t) \) under these conditions.

Theorem 3

Under the condition that \( \{C_j(p_j^k) - C_j(\tilde{u}_j^k, \tilde{p}_j^k, t)\} \) is a nonincreasing sequence in k, once a unit has been selected in Step 2 in the generic unit decommitment algorithm at some iteration, it will not be selected again in Step 2 at any future iteration. So the decommitment algorithm terminates within I iterations, where \( I = \frac{\text{number of units}}{\text{number of iterations}} \).

Proof: Suppose unit j is selected at iteration \( k' \), and its tentative commitment of unit k is \( \hat{u}_j^k \), then \( \hat{u}_j^{k+1} \) is
\[ \sum_{i=1}^{T} \left[ (C_j(\tilde{p}_i^k) + S_j(\tilde{u}_j^k, t)) \hat{u}_j^k + \Delta C_j(\tilde{u}_j^k, \tilde{p}_j^k, t)(1 - \hat{u}_j^k) \right] \]
\[ \leq \sum_{i=1}^{T} \left[ (C_j(\tilde{p}_i^k) + S_j(\tilde{u}_j^k, t)) \hat{u}_j^k + \Delta C_j(\tilde{u}_j^k, \tilde{p}_j^k, t)(1 - \hat{u}_j^k) \right] \]
for any \( \{u_j^k\} \) satisfying equation (13).

IV. Approximate methods

In this section, we discuss methods to approximate \( \Delta C_j(\tilde{u}_j^k, \tilde{p}_j^k, t) \) in the objective function of (P). As previously mentioned, \( \Delta C_j(\tilde{u}_j^k, \tilde{p}_j^k, t) \) is an estimate of the increased generating costs of all on-line units other than unit j due to
its decommitment. The exact value of $\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t)$ is given in equation (12). To solve equation (12), the equivalent of an extra economic dispatch has to be solved, i.e. $\text{edp}(\bar{J}_I\backslash\{j\}, t)$, for $\forall t$. Therefore at each iteration of the decommitment algorithm $I + 1$ economic dispatches are performed. If each unit is only selected once, as discussed at the end of the previous section, there are at most $I$ iterations, and the algorithm will require performance of no more than $I + I(I + 1)/2$ economic dispatches in total. Even though the economic dispatch can be efficiently solved (e.g. [7,15]), in a large-scale system it is desirable to approximate $\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t)$ without performing an extra economic dispatch. Next we discuss two methods to approximate $\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t)$.

IV.1 Guaranteed descent method
If $\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t)$ is approximated, a sufficient condition to guarantee that solving $(P_j)$ with respect to $(\hat{\mathbf{u}}, \hat{\mathbf{p}})$ can yield a commitment no worse than $\hat{\mathbf{u}}$ is that $\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t)$ satisfies

$$\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t) \geq \text{edp}(\bar{J}_I\backslash\{j\}, t) - (\text{edp}(\bar{J}_I, t) - C_j(\hat{p}_{ji}, \bar{u}_{ji}))$$

(24)

That is, $\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t)$ should be overestimated. Since the last two terms (in parentheses) in equation (24) are fixed and known when solving $(P_j)$, the exact value of $\Delta C_j$ given in equation (12) can be regarded as the value of the solution of a minimization problem, i.e., the economic dispatch $\text{edp}(\bar{J}_I\backslash\{j\}, t)$. To overestimate the solution value of a minimization problem, any feasible solution of the minimization problem will do. So any feasible dispatch among the unit index set $\bar{J}_I\backslash\{j\}$ can be applied to satisfy equation (24). An easy way to create a feasible and reasonably good dispatch is to apply a priority list. All the units are assigned a priority order which is determined by the slope of the incremental cost curve $2a_{ij}/t$ of the unit. At time $t$, if unit $i = j$ is to be decommitted, its generation amount $\hat{p}_{ji}$ will be distributed to other on-line units. All the on-line units other than unit $j$ increase their outputs in the order of the priority list, subject to unit capacity constraint, until $\hat{p}_{ji}$ is totally replaced. The algorithm used to overestimate the exact value of $\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t)$ by the priority list is given below.

Guaranteed descent method: by priority list
Step 0: $d \leftarrow \hat{p}_{ji}; J \leftarrow \bar{J}_I\backslash\{j\}; \Delta C \leftarrow 0$

Step 1: Let $l = \arg \min_{i \in J} a_{il}$. Obtain the following:

$$\hat{p}_{li} = \min\{\hat{p}_{li} + d, a_{il} \hat{p}_{li}^{\text{max}}\}$$

(25)

$$d \leftarrow d - (\hat{p}_{li} - \hat{p}_{li})$$

(26)

and

$$\Delta C \leftarrow \Delta C + C_l(\hat{p}_{li}) - C_l(\hat{p}_{li})$$

$$= \Delta C + a_{il}(\hat{p}_{li} - \hat{p}_{li}) + a_{il}(\hat{p}_{li}^2 - \hat{p}_{li}^2)$$

(27)

Step 2: If $d = 0$, stop and $\Delta C$ is the estimation; otherwise $J \leftarrow \bar{J}_I\backslash\{l\}$ and go to Step 1. ■

The priority list algorithm above is very efficient. It obtains a good approximation of $\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t)$ based on our simulation experience.

IV.2 First order approximate method
Another method of approximating $\Delta C_j$ obtains a first order approximation. Assume that solving $\text{edp}(\bar{J}_I\backslash\{j\}, t)$ yields generation level $\hat{p}_{ji} + \Delta \hat{p}_{ji}$ for unit $i \in \bar{J}_I\backslash\{j\}$ such that

$$\sum_{i \in \bar{J}_I\backslash\{j\}} \Delta \hat{p}_{ji} = \hat{p}_{ji}.$$ 

Since all the fuel cost functions are assumed to be smooth, from equation (7), we have

$$\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t) = \sum_{i \in \bar{J}_I\backslash\{j\}} C_i(\hat{p}_{ji}) + \Delta \hat{p}_{ji} - C_i(\hat{p}_{ji})$$

$$= \sum_{i \in \bar{J}_I\backslash\{j\}} C_i(\hat{p}_{ji}) \Delta \hat{p}_{ji}$$

$$= \tilde{\lambda}_i \sum_{i \in \bar{J}_I\backslash\{j\}} \Delta \hat{p}_{ji} = \tilde{\lambda}_i \hat{p}_{ji}$$

(28)

where $\tilde{\lambda}_i$ is the marginal cost at hour $t$ obtained in solving $\text{edp}(\bar{J}_I)$. Note that in equation (28) we apply the optimality condition of economic dispatch, i.e. operating all units at the same marginal cost. However, this is only an approximation (cf. equation (10)) of the incremental costs of generating units when unit capacity constraints are present.

Also by Lemma 1 we know that the first order approximate method always underestimates the exact value of $\Delta C_j(\hat{\mathbf{u}}, \hat{\mathbf{p}}, t)$. Therefore the selected tentative commitment does not necessarily guarantee improvement. This sometimes causes the algorithm to terminate prematurely. On the other hand, since Lemma 2 (v) implies that the condition stated in Theorem 3 is satisfied for the first order approximation method, we have the following corollary.

Corollary 4
When applying the first order approximation method to evaluate $\Delta C_j$, the unit decommitment algorithm terminates within $I$ iterations, where $I$ is the number of the units, independent of the selection rule used in Step 2 in the algorithm.

After substituting $\Delta C_j$ in equation (13) by $\tilde{\lambda}_i \hat{p}_{ji}$ and rearranging terms, the objective of $(P_j)$ is equal to the following:

$$\min_{\hat{u}_{ji}, \hat{p}_{ji}} \sum_{i = 1}^{T} \left[ C_j(\hat{p}_{ji}) u_{ji} - \tilde{\lambda}_i \hat{p}_{ji} u_{ji} + S_j(\hat{u}_{ji}, t) + \sum_{i = 1}^{T} \tilde{\lambda}_i \hat{p}_{ji} t \right]$$

(29)

The last term in equation (29) is a constant and can be ignored in the objective. Since $\tilde{\lambda}_i$ is the marginal cost obtained from economic dispatch, we have the following proposition.

Proposition 5
The following problem $(\hat{P}_j)$ yields the same commitment solution as $(P_j)$ with respect to $(\hat{\mathbf{u}}, \hat{\mathbf{p}})$:

$$(\hat{P}_j) \min_{u_{ji}, \hat{p}_{ji}} \sum_{i = 1}^{T} [C_j(\hat{p}_{ji}) u_{ji} - \tilde{\lambda}_i \hat{p}_{ji} u_{ji} + S_j(\hat{u}_{ji}, t)]$$

subject to equations (3)–(6) and (13), where $[\lambda_e]$ are associated with $(\hat{\mathbf{u}}, \hat{\mathbf{p}})$, satisfying equation (10).

Proof: Since $\hat{p}_{ji}$ and $\tilde{\lambda}_i$ satisfy equation (10), when $u_{ji} = 1$, $\hat{p}_{ji}$ solves $\min [C_j(\hat{p}_{ji}) - \tilde{\lambda}_i \hat{p}_{ji}]$ subject to the unit capacity constraint $\hat{p}_{ji}^{\text{min}} \leq \hat{p}_{ji} \leq \hat{p}_{ji}^{\text{max}}$. ■

$(\hat{P}_j)$ resembles a unit subproblem in the Lagrangian relaxation approach for solving the unit commitment problem, but in which only the system load equations are relaxed, and the values of the Lagrangian multipliers are taken from economic dispatch. This was also the basic idea behind the method in [13]. The method in [13] initially turns on as many units as possible. It solves problem $(P_j)$ but only subject to
equations (4)–(6), and repeatedly updates the problem in the manner of the decommitment algorithm proposed in Section III of this paper. Since originally all units were turned on, any improvement in the commitment through \( \hat{P}_j \) can only involve unit decommitment.

V. Discussion

V.1 Optical unit decommitment

The proposed unit decommitment algorithm determines an optimal strategy to improve by decommitting a single unit at each iteration. Similarly, an optimal strategy to improve by two units at each iteration can be devised. In that case, in each hour of the planning horizon, at most, four combinations of the corresponding on-line units are considered for decommitment. If solved by dynamic programming, this requires extending the state space to include these combinations of on-line units. Intuitively, an optimal unit decommitment in an hour considers all possible combinations of all on-line units to be decommitted and determines the optimal strategy, which itself is a constrained unit commitment problem. Therefore, the optimal unit decommitment problem is a difficult combinatorial problem. For a detailed discussion, the interested reader is directed to [1]. In our design, the unit decommitment algorithm is a post-processing method to aid the two-phase (dual optimization phase and feasibility phase) Lagrangian relaxation method in improving solution quality. Given a feasible solution by the Lagrangian relaxation method, it is believed that the possibility of decommitting more than one unit at a time without affecting the feasibility is relatively rare.

V.2 Solving unit commitment by unit decommitment

When applied to solve a complete unit commitment problem with all the available units turned on at all hours initially, our unit decommitment algorithm falls into the class of methods proposed in [13]. When the system is thus overcommitted (in terms of surplus between the system capacity and system demand to measure the possible combinations of unit decommitment), it is not clear whether improving one unit at a time with all other units fixed is a near-optimal strategy to improve the current unit commitment. Recently we have conducted extensive testing on the unit decommitment method proposed herein as a unit commitment algorithm, and compared it with a Lagrangian relaxation method. The test results suggest that this method could be an efficient way to obtain a reasonably good solution of the unit commitment problem [1].

VI. Numerical results

The decommitment algorithms proposed in this paper have been implemented in FORTRAN on an HP 700 workstation. We compare the performance of the decommitment algorithm (DA) to that of its two approximate variants, the guaranteed descent method (DA-GD) and the first order method (DA-FO) introduced in Section IV. For our numerical tests of these three algorithms, we use randomly generated instances of the unit commitment problem. In other words, we randomly generate the parameters for the problem formulation described by equations (1)–(6) according to Table 1. All three methods use the steepest descent type of selection rule in Step 2 of the algorithm. Eight types of system with different numbers of units and lengths of planning horizon are tested. Cases are denoted by (no. of units × no. of hours in planning horizon): (10 × 24), (10 × 168), (20 × 24), (20 × 168), (30 × 24), (30 × 168), (40 × 24) and (40 × 168). Unit commitment instances of each case are randomly generated as follows. Let \( \text{rand}(x, y) \) denote a random number generator which generates numbers uniformly distributed between \( x \) and \( y \). Using this distribution we obtain the randomized parameter values for the unit commitment formulation as given in Table 1.

The given initial feasible commitment \( \bar{u} \) is generated uniformly among \([0, 1]\) in succession, starting from the first hour, while the minimum uptime and downtime constraints are satisfied. Note that the demand \( D_i \) is generated after \( \bar{u} \) to guarantee equation (11) is satisfied.

When testing each case, 100 instances are generated based on the configuration in Table 1. For each instance the three algorithms are applied. Since each instance is different, only the relative performances of the three algorithms are recorded, including the relative total generating costs (in Table 2) and the relative CPU time (in Table 3). However, we also provide the average cost and CPU time of the 100 instances of each case solved by the DA in Tables 2 and 3 respectively. Note that in the following tables, the two numbers inside the parentheses under an averaged number indicate the range of all the corresponding sample points obtained during the testing of all instances.

<table>
<thead>
<tr>
<th>Case</th>
<th>DA</th>
<th>DA-GD</th>
<th>DA-FO</th>
<th>Average cost of DA (( $10^5 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 × 24</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>(0.978–1.011)</td>
</tr>
<tr>
<td>10 × 168</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>(0.978–1.011)</td>
</tr>
<tr>
<td>20 × 24</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>(0.978–1.011)</td>
</tr>
<tr>
<td>20 × 168</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>(0.978–1.011)</td>
</tr>
<tr>
<td>30 × 24</td>
<td>0.998</td>
<td>0.998</td>
<td>0.998</td>
<td>(0.978–1.011)</td>
</tr>
<tr>
<td>30 × 168</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>(0.978–1.011)</td>
</tr>
<tr>
<td>40 × 24</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>(0.978–1.011)</td>
</tr>
<tr>
<td>40 × 168</td>
<td>0.998</td>
<td>0.998</td>
<td>0.998</td>
<td>(0.978–1.011)</td>
</tr>
</tbody>
</table>

Table 3. Average relative CPU time

<table>
<thead>
<tr>
<th>Case</th>
<th>DA</th>
<th>DA-GD</th>
<th>DA-FO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU time of DA (s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 × 24</td>
<td>0.717</td>
<td>0.670</td>
<td>(0.500–0.727)</td>
</tr>
<tr>
<td>10 × 168</td>
<td>0.732</td>
<td>0.643</td>
<td>(0.995–1.018)</td>
</tr>
<tr>
<td>20 × 24</td>
<td>0.510</td>
<td>0.467</td>
<td>(0.416–0.540)</td>
</tr>
<tr>
<td>20 × 168</td>
<td>0.541</td>
<td>0.472</td>
<td>(0.426–0.513)</td>
</tr>
<tr>
<td>30 × 24</td>
<td>0.401</td>
<td>0.356</td>
<td>(0.319–0.370)</td>
</tr>
<tr>
<td>30 × 168</td>
<td>0.419</td>
<td>0.359</td>
<td>(0.355–0.427)</td>
</tr>
<tr>
<td>40 × 24</td>
<td>0.340</td>
<td>0.298</td>
<td>(0.250–0.312)</td>
</tr>
<tr>
<td>40 × 168</td>
<td>0.348</td>
<td>0.298</td>
<td>(0.305–0.337)</td>
</tr>
</tbody>
</table>

Tables 2 and 3 show that the DA-GD and the DA-FO methods are both good approximations for the DA. In terms of cost saving, DA-GD, on average, obtains better solutions than DA and DA-FO. The differences between the solutions obtained by these three methods are generally within 2%, based on our testing. It is clear that both approximate methods, DA-GD and DA-FO, require much less computational time than DA, and require less CPU time as the system gets larger. DA-GD requires a sorting procedure to determine the priority list, and therefore always takes more time than DA-FO. Based on the comparison and our experience, we recommend the DA-GD method because of its guaranteed descent property; on the other hand without this property DA-FO in many cases will terminate in the first few iterations due to a poorer estimate of \( \Delta C_i(u, \bar{B}, I) \) in equation (12) as discussed in Section III.1.

Table 4 records the number of iterations required by these three methods on the test cases. The result justifies the convergence analysis in Section III.1.

As aforementioned, the unit decommitment method is proposed as a post-processing module for unit commitment algorithms. We integrate the three methods into a Lagrangian relaxation (LR) algorithm, and examine how the unit decommitment algorithm can improve the solution of the LR approach. A 30-unit-168-hour unit commitment instance is solved using the LR algorithm with different stopping criteria. For the purposes of illustration, the stopping criterion adopted here is the number of iterations. The test result is given in Table 5. In the test, we operate the LR dual optimization and terminate it at a given number of iterations. The dual objective value is recorded in Table 5 in the first row in each section. A feasibility phase follows to obtain a feasible solution, whose corresponding total generating cost is recorded in the second row of each section in Table 5. Based on this feasible solution, we apply the DA-GD algorithm to improve the feasible solution and obtain the improved generating cost recorded in the third row. The duality gaps corresponding to the solutions before and after applying the DA-GD algorithm are also given in Table 5. In the column under CPU time, in each section, the upper number is the CPU time required by a complete LR algorithm. The lower number indicates only the CPU time required by the DA-GD.

In Table 5, one can see that the dual optimum is approached between the 45th and the 50th iteration. Because of the highly nonlinear behavior of the feasibility phase, it is not clear when to terminate the dual optimization followed by a feasibility phase to yield the best result. Of course, duality theory suggests fully solving the dual optimization. Comparing the two cases of terminating the dual optimization at the 40th and the 45th iteration, the latter with its higher dual objective value yields a poorer solution after the

Table 4. Number of iterations

<table>
<thead>
<tr>
<th>Case</th>
<th>DA</th>
<th>DA-GD</th>
<th>DA-FO</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 × 24</td>
<td>6.01</td>
<td>5.88</td>
<td>5.66</td>
</tr>
<tr>
<td>(4–9)</td>
<td>(3–9)</td>
<td>(3–9)</td>
<td></td>
</tr>
<tr>
<td>10 × 168</td>
<td>8.83</td>
<td>8.87</td>
<td>7.81</td>
</tr>
<tr>
<td>(6–10)</td>
<td>(6–10)</td>
<td>(5–10)</td>
<td></td>
</tr>
<tr>
<td>20 × 24</td>
<td>10.91</td>
<td>10.71</td>
<td>10.46</td>
</tr>
<tr>
<td>(7–15)</td>
<td>(7–14)</td>
<td>(7–14)</td>
<td></td>
</tr>
<tr>
<td>20 × 168</td>
<td>16.14</td>
<td>15.83</td>
<td>14.65</td>
</tr>
<tr>
<td>(12–19)</td>
<td>(13–19)</td>
<td>(10–18)</td>
<td></td>
</tr>
<tr>
<td>30 × 24</td>
<td>16.03</td>
<td>15.66</td>
<td>15.32</td>
</tr>
<tr>
<td>(11–21)</td>
<td>(10–20)</td>
<td>(9–19)</td>
<td></td>
</tr>
<tr>
<td>30 × 168</td>
<td>23.34</td>
<td>22.91</td>
<td>21.54</td>
</tr>
<tr>
<td>40 × 24</td>
<td>20.49</td>
<td>20.04</td>
<td>19.88</td>
</tr>
<tr>
<td>(15–26)</td>
<td>(15–25)</td>
<td>(12–25)</td>
<td></td>
</tr>
<tr>
<td>40 × 168</td>
<td>30.33</td>
<td>29.48</td>
<td>28.51</td>
</tr>
</tbody>
</table>

Table 5. Comparison of methods for unit commitment

<table>
<thead>
<tr>
<th>No. of iterations</th>
<th>Total cost ($)</th>
<th>Duality gap (%)</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Dual Objective</td>
<td>9,252,032</td>
<td>0.53</td>
</tr>
<tr>
<td>DA-GD</td>
<td>9,300,814</td>
<td>0.59</td>
<td>8.40</td>
</tr>
<tr>
<td>Feasibility Phase</td>
<td>9,288,790</td>
<td>0.39</td>
<td>8.40</td>
</tr>
<tr>
<td>25</td>
<td>Dual Objective</td>
<td>9,254,664</td>
<td>0.44</td>
</tr>
<tr>
<td>Feasibility Phase</td>
<td>9,295,743</td>
<td>0.44</td>
<td>18.36</td>
</tr>
<tr>
<td>DA-GD</td>
<td>9,287,256</td>
<td>0.35</td>
<td>5.2</td>
</tr>
<tr>
<td>30</td>
<td>Dual Objective</td>
<td>9,255,323</td>
<td>0.43</td>
</tr>
<tr>
<td>Feasibility Phase</td>
<td>9,295,024</td>
<td>0.43</td>
<td>22.11</td>
</tr>
<tr>
<td>DA-GD</td>
<td>9,287,479</td>
<td>0.35</td>
<td>8.84</td>
</tr>
<tr>
<td>35</td>
<td>Dual Objective</td>
<td>9,255,988</td>
<td>0.39</td>
</tr>
<tr>
<td>Feasibility Phase</td>
<td>9,292,481</td>
<td>0.39</td>
<td>23.57</td>
</tr>
<tr>
<td>DA-GD</td>
<td>9,288,322</td>
<td>0.35</td>
<td>7.90</td>
</tr>
<tr>
<td>40</td>
<td>Dual Objective</td>
<td>9,256,526</td>
<td>0.43</td>
</tr>
<tr>
<td>Feasibility Phase</td>
<td>9,296,280</td>
<td>0.43</td>
<td>23.27</td>
</tr>
<tr>
<td>DA-GD</td>
<td>9,280,256</td>
<td>0.26</td>
<td>6.54</td>
</tr>
<tr>
<td>45</td>
<td>Dual Objective</td>
<td>9,257,763</td>
<td>0.68</td>
</tr>
<tr>
<td>Feasibility Phase</td>
<td>9,320,547</td>
<td>0.68</td>
<td>25.78</td>
</tr>
<tr>
<td>DA-GD</td>
<td>9,281,954</td>
<td>0.26</td>
<td>7.24</td>
</tr>
<tr>
<td>50</td>
<td>Dual Objective</td>
<td>9,257,659</td>
<td>0.37</td>
</tr>
<tr>
<td>Feasibility Phase</td>
<td>9,292,240</td>
<td>0.37</td>
<td>26.46</td>
</tr>
<tr>
<td>DA-GD</td>
<td>9,280,092</td>
<td>0.24</td>
<td>7.42</td>
</tr>
<tr>
<td>DA-FO</td>
<td>9,395,589</td>
<td>0.24</td>
<td>40.56</td>
</tr>
<tr>
<td>DA-GD</td>
<td>9,382,742</td>
<td>0.24</td>
<td>21.28</td>
</tr>
<tr>
<td>DA-PO</td>
<td>9,433,649</td>
<td>0.24</td>
<td>20.65</td>
</tr>
</tbody>
</table>
feasibility phase. The poor quality of this solution implies overcommitment. After applying DA-GD, the duality gap is reduced from 0.68 to 0.26%. From Table 5, it can also be seen that the unit decommitment algorithm generally relieves the unpredictable effects of the heuristic feasibility phase, and makes the solution less sensitive to the number of iterations used as the stopping criterion in the dual optimization. Also note that, with unit decommitment, it may be a good strategy to moderately overcommit generating units in the feasibility phase. In terms of the CPU time required for obtaining a feasible solution, Table 5 suggests a possibly advantageous strategy of applying the (cheap) unit decommitment algorithm to substitute for the (expensive) dual optimization. Also, by determining how much saving could be achieved by unit decommitment, our method can be used as a tool to measure the solution quality obtained by a unit commitment algorithm.

In the bottom section of Table 5, the three algorithms are applied as a complete unit commitment algorithm, with all the units initially turned on to the greatest extent without violating the minimum up time and downtime constraints. As discussed in Section V, the approximate method (DA-GD) efficiently obtains a reasonably good solution compared with regular unit commitment algorithms like LR.

VII. Conclusions

In this paper we present a unit decommitment method for power system scheduling. Given a feasible unit commitment, our algorithm determines an optimal strategy for decommitting overcommitted units based on dynamic programming. The method was developed as a post-processing tool to aid the existing unit commitment algorithm in improving solution quality. Two approximate methods are proposed. In our numerical test, we show that the two approximate methods are much faster than the proposed decommitment algorithm, and the difference between the solutions obtained by the proposed decommitment algorithm and the two approximate algorithms is within 2%, on average. We also have integrated one approximate algorithm into a Lagrangian relaxation algorithm. The numerical results show that with unit decommitment, solution quality can be generally improved efficiently. Unit decommitment also relieves the unpredictable effects of heuristics and makes the Lagrangian relaxation approach more robust. The method of unit decommitment presented in this paper has been extended to accommodate more complicated constraints including ramp constraints and transmission constraints [1]. We shall present this extension in a future paper.

VIII. Acknowledgements

This work was partly supported by the National Science Foundation under Grant IRI-9120074 and by PG&E’s R&D Department. Their support is greatly appreciated.

IX. References


