DYNAMIC PROGRAMMING BASED UNIT SCHEDULING: A FEASIBILITY STUDY

Y. AL KALAANI
Northern Illinois University, College of Engineering & Engineering Technology, Technology Department, Still Hall 603D, Dekalb, Illinois 60115 alkalaan@ceet.niu.edu

(Received April 2005 and accepted October 2005)

In dynamic programming-based power scheduling algorithms, thousands of hourly economic dispatches must be performed to consider every possible unit combination over all the stages of the optimization interval. If the unit commitment problem is constrained to observe a minimum system spinning reserve and an economic dispatch of a combination of units does not comply with this requirement, necessary and sufficient conditions have been established to guarantee that the dispatch of these units will meet the constraint. However, these conditions can only be checked for feasibility after a dispatch is performed. In this paper, we present necessary and sufficient conditions for the feasibility of unit combinations that can be checked off-line. That is, before the start of the unit commitment algorithm, and thus before any economic dispatches are performed, thereby rendering a very efficient unit scheduling algorithm in terms of computer memory and execution time. Moreover, these feasibility conditions are independent of the problem formulation and thus can easily be applied to other unit commitment algorithms. Examples are provided to illustrate the efficiency attainable by the implementation of these conditions.

1. INTRODUCTION

Unit commitment is one of the decision-making levels in the hierarchy of power system operations management. The optimization problem is posed over time horizons that vary from 24 hours to a week. The objective is to determine the set of generating units, among those owned by a utility that should be connected to the power grid on a hourly basis to supply the required energy at minimum operating cost over the scheduling horizon. This optimization problem is constrained by the unit characteristics and the constraints. Since the objective of the unit commitment is to determine a cyclic schedule that will meet the system constraints at minimum cost, the economic operation of a power system may be formulated as a dynamic optimization problem. The problem is dynamic in the sense that decisions to startup and/or shutdown units at any stage cannot be made without considering the states of the system at some other stages.

Many mathematical programming techniques proposed to solve the unit commitment problem. Dynamic programming is the only one that always yields a globally optimal solution because it can operate upon nonlinear, non-convex, integer and otherwise non-differentiable functions not amenable to other optimization techniques. Although dynamic programming-based solution algorithms provide optimal commitment schedules, execution time and memory requirements are affected by the "curse of dimensionality," that is by the number of unit combinations to be considered in the solution process. Generally, unit commitment is a computationally intensive algorithm, since thousands of hourly economic dispatches must be performed to consider every possible unit combination over all the stages in the optimization interval. Therefore, any effort toward the elimination of any unfeasible unit combinations performed off-line, that is before the start of the commitment algorithm and thus before any dispatches are considered, will handsomely

15
pay off in the efficient use of computer resources. When
the commitment problem is constrained to observe a
minimum system spinning reserve and a dispatch of a
unit combination does not comply with the requirement,
the problem may still be solvable by the re-dispatch of
these units, as long as the conditions prescribed by W.G.
Wood⁶⁰ are satisfied. However, these feasibility
conditions can only be checked after a dispatch has been
performed and, as a consequence, many dispatches that
turn out to be unfeasible are needlessly computed in the
process.

In this paper we present the off-line conditions that
a unit combination must satisfy to not only meet the
minimum system spinning reserve constraint, but also
the power balance constraint, the unit capacity limits,
and all the other pertinent constraints as well. These
conditions are shown to be necessary. We further show
that these conditions turn out to be both necessary and
sufficient when the spinning reserve constraint is to be met by re-dispatch. Clearly then,
when these conditions are not fulfilled by a unit
combination, that combination can be discarded as
unfeasible by the scheduling algorithm, thereby
reducing drastically the number of decisions to be
considered in the solution space. The net result is a
very efficient commitment algorithm as illustrated by
easy examples. Furthermore, these feasibility conditions are
independent of the problem formulation and thus can
easily be implemented in other unit commitment
algorithms.

2. PROBLEM FORMULATION

To mathematically state the constrained unit
commitment problem, we need the following definitions

Definition 1.

Let the time horizon be divided into \( M \) hourly stages. Then
the set of all stages is \( K = \{0, 1, 2, ..., k, ..., M\} \). Assume
that a utility has \( N \) generating units. Then the set
of all units, or the universal set, is \( I = \{1, 2, ..., i, ..., \ N\} \). Since a generating unit may be on or off at stage
\( k \), let the status of unit \( i \), or commitment variable, be
defined by the binary set \( d_{ik} = \{0, 1\} \).

At any stage \( k \), the set of all units may be divided
into two mutually exclusive sets:

- \( I_{ON} \in \left\{ i \in I \mid d_{ik} = 1 \right\} \), the set of units on-line at stage
  \( k \), and
- \( I_{OFF} \in \left\{ i \in I \mid d_{ik} = 0 \right\} \), the set of units off-line at stage
  \( k \), that is, satisfying \( I_{ON} \cup I_{OFF} = I \) and \( I_{ON} \cap I_{OFF} = \phi \). Since the status of any unit is described by a digit
  in the binary set, then the set of all unit combinations at
  stage \( k \) is given by \( J_k = \{1, j_2, ..., j_k, ..., J_M\} \). Then, the set of
  all possible unit combinations over the entire time
  horizon \( M \) is defined as \( J = \{J1, J2, ..., J_k, ..., J_M\} \).

Given the status \( d_{ik} \) of every unit \( i \in I \) at any stage \( k \), the
unit combination number at that stage may readily be
obtained by the binary to decimal transformation,

\[
I_k = \sum 2^{i-1}d_{ik} = \sum 2^{i-1} \quad k \in K, j_k \in J_k
\]

If \( p_k \) is the power allocated to each unit \( i \) in any unit
combination \( j_k \), then the production cost for this
combination is

\[
PCOST( j_k ) = \sum_{i=1}^{N} C_i ( p_{ik} ) di_k
\]

where \( C_i(\cdot) \) is the production cost \([$/h]\) of unit \( i \) at stage
\( k \).

The transitional cost associated with the start-up and
shut-down of each unit \( i \) in unit combination \( j_k \) relative to
unit combination at previous stage \( j_{k-1} \) is

\[
TCOST( j_{k-1}, j_k) = \sum_{i=1}^{N} |di_k - di_{k-1}| TC_i(t_i, k-1)
\]

where \( TC_i(\cdot) \) is the transitional cost \([$/h]\) of unit \( i \) from stage \( k-1 \) to \( k \), and \( t_i, k-1 \) is the number of hours that unit \( i \) has been on or off up to
stage \( k-1 \). Then the optimization function of the unit
commitment problem can be stated as

\[
\arg \min Z(J) = \sum_{k=1}^{K} (PCOST( j_k ) + TCOST
\quad (j_{k-1}, j_k)) \quad \forall j_k \in J_k \quad \forall J_k \in J
\]

where \( Z(J) \) is the total cost of operation \([$/\text{h}]\) during the
entire time horizon \( M \).

This optimization problem is subject to many
constraints. Of interest here are the following:

1. Power Balance Constraint.

\[
PG(j_k) = PD_k \ [MW] \ \forall j, \ \forall k
\]

where

\[
PG(j_k) = \sum_{i=1}^{N} p_{ik} dik
\]

is the total power generated by unit combination \( j \) at
stage \( k \), and \( p_{ik} \) is the generation allocated to unit \( i \) in
that combination, and \( PD_k \) is the system demand plus
transmission losses at any given stage \( k \).

2. Unit Capacity Constraints.

\[
P_{i, \text{MIN}} \leq p_{ik} \leq P_{i, \text{MAX}} \ [MW] \ \forall i, \ \forall k
\]

3. Spinning Reserve Constraint.

\[
SSR(j_k) \geq MSSR \ [MW] \ \forall k, \ \forall j
\]

where

\[
SSR(j_k) = \sum_{i \in I_{ON}} \min (P_{i, \text{MAX}} - p_{ik}, MSR_i)
\]

MSR is the maximum spinning reserve for unit \( i \),
MSSR is the minimum system spinning reserve.
2.1. Dynamic Programming

Dynamic programming is a powerful mathematical tool that utilizes the principle of optimality to solve optimization problems that can be characterized by sequential decision processes. First introduced by Dr. Richard Bellman in the late 1950's it has since been widely used in the allocation of scarce resources, especially in the fields of economics and operation research, because it can yield optimal solutions. Dynamic programming is considered an effective solution technique in power systems, not only because scheduling is naturally a sequential decision process, but also because formulation of the unit commitment problem results in a non-linear, non-convex, time dependent, and mix-integer problem that is not easily amenable to other solution techniques.

In dynamic programming based unit commitment algorithms, as shown in Figure 1, for each time interval (usually an hour), different combinations of units, which render feasible solutions to the scheduling problem, are considered. At each stage, economic dispatch is performed on every feasible unit combination to calculate its generation at equal fuel incremental costs. Taking into account transitional costs associated with the units’ startup and shutdown, the algorithm could proceed in a forward direction to cover the entire scheduling horizon. At each stage, a pointer is assigned to every feasible unit combination that uniquely identifies its predecessor yielding the lowest cumulative cost. The optimal schedule is obtained by tracing the path linking the successive decisions that rendered the least total cumulative cost. Since transitional costs are time dependent, forward dynamic programming must be used.

Because of its combinatorial nature, dynamic programming suffers from exponential increase of dimensionality which can prevent its applications in large-scale systems. For instance, for a time horizon of M stages and N generating units, there are a total of \((2^N-1)^M\) possible unit combinations that the dynamic programming algorithm must consider during the scheduling period. The exponential increase in the number of combinations can quickly result in huge computational time and memory requirements.

Therefore, any effort toward the elimination of infeasible unit combinations, or states, performed offline, that is before the start of the commitment algorithm and thus before any dispatched are considered, will handsomely pay off in the efficient use of computer resources. In this paper, closed-form dispatch strategies and state feasibility conditions are established to eliminate infeasible unit combinations thus rendering a very efficient commitment algorithm.

The state transfer or recursion function required to solve this optimization problem using dynamic programming is given by

\[
CCOST(jk) = \min (CCOST(jk-1) + TCOST(jk-1, jk) + PCOST(jk) \forall jk-1 \in Jk-1, jk \in Jk, k \in K).
\]

where \(CCOST(.)\) is the cumulative cost \(S\) associated with every state \(j_k\), for all states \(j_{k-1}\), subject to all the problem constraints.

3. FEASIBILITY CONDITIONS

If constraints (1)-(3) are not met by a unit combination at a stage in the solution of the commitment problem, that unit combination is declared infeasible and is no longer retained in the solution process. The fulfillment of these constraints may be tested after the economic dispatch of each unit combination is performed. Since economic dispatch is another optimization problem embedded in the unit commitment algorithm, and since for \(N\) units there are \(2^N-1\) possible combinations at every stage \(k\), then thousands of dispatches over the entire time horizon (a day to a week) would need to be performed in the solution process just to discover that a great many of them were not feasible. Therefore, such an \(a\ posteriori\) test is very inefficient in terms of computer memory and time. Furthermore, an infeasible unit combination, which meets constraint (1) and (2) but not (3), may be rendered feasible by the re-dispatch of the units involved in that combination whenever certain feasibility conditions are satisfied. But these conditions can only be tested after a dispatch has been performed. What is clearly needed is an \(a\ priori\) test for feasibility that is performed off-line. That is, a test which eliminates those unit combinations that are guaranteed to be infeasible before the start of the unit commitment algorithm, and thus before any dispatches are performed. To this end, the following definitions are needed.

Definition 2.

A unit combination whose on-line units form the set \(I_{ON}\) is said to be feasible when the solution of the economic dispatch problem satisfies constraints (1)-(3).

Definition 3.

Consider a set of on-line units \(I_{ON} = I_1 \cup I_2\) where

\[
I_1 = \{ i \in I_{ON} | P_{i,\text{MAX}} - P_{i,k} \geq MSR \},
\]

and

\[
I_2 = \{ i \in I_{ON} | P_{i,\text{MAX}} - P_{i,k} < MSR \}.
\]
It is clear from (5) and (6) that \( I_2 \cap I_2 = \emptyset \). However, either one of these two subsets may be empty. We can now state the following Lemmas.

**Lemma 4.**
If a set \( I_{ON} \) is feasible at stage \( k \), then it satisfies
\[
\sum_{i \in I_{ON}} MSR_i \geq MSSR \quad i \in I_{ON}
\]

(7)

**Proof:** From (4) it is clear that
\[
SSR( jk ) \leq \sum_{i \in I_{ON}} MSR_i \quad i \in I_{ON}
\]

and by (3), condition (5) follows immediately.

**Lemma 5.**
If a set \( I_{ON} \) is feasible at stage \( k \) then it satisfies
\[
\sum_{i \in I_{ON}} P_i, \text{MAX} \leq PD_k \leq \sum_{i \in I_{ON}} P_i, \text{MAX} - MSSR
\]

(8)

**Proof:** Provided in Appendix A

The conditions provided by Lemmas 4 and 5, are both necessary, and can be tested a priori. That is, when at any stage a unit combination violates either one of them, that unit combination can be eliminated off-line and it will not be considered by the commitment algorithm and, thus, there will be no need to perform a dispatch for it either.

**Corollary 6.**
If at any stage \( k \) conditions (7) and (8) are violated for \( I_{ON} = I \), then the unit commitment problem has no solution whatsoever.

**Proof:** Follows trivially from Lemmas 4 and 5.

Although the conditions of this corollary may rarely occur if ever, it should nevertheless be tested since the commitment algorithm may be used for planning studies. Clearly, this test should precede those indicated by Lemmas 4 and 5.

**Lemma 7.**
If a set \( I_{ON} \) = \( I_1 \cup I_2 \) satisfying (7) and (8) is not feasible by violating (3) then \( I_1 \neq \emptyset \) and \( I_2 \neq \emptyset \)

**Proof:** Assume condition (3) is violated with \( I_{ON} = I_1 \).

By (4), their total contribution to the reserve is
\[
SSR( jk ) = \sum_{i \in I_{ON}} MSR_i
\]

But by Lemma 4, condition (3) is satisfied, leading to a contradiction. A similar contradiction results if \( I_{ON} = I_2 \) instead, since by the right hand side of Lemma 5
\[
\sum_{i \in I_{ON}} P_i, \text{MAX} - PD_k \geq MSSR
\]

There are instances in which the economic dispatch of a set \( I_{ON} \), while satisfying constraints (1) and (2), will violate condition (3). However, a re-dispatch of these units may render compliance with condition (3). However, instead of checking the feasibility conditions for a re-dispatch after the results of an economic dispatch are obtained, we now present our main result in theorem form. Briefly, this shows that conditions (7) and (8), which can be checked a priori, are both necessary and sufficient to guarantee the feasibility of a re-dispatch. But first we need the following definitions.

**Definition 8.**
When a unit combination \( j_k \) contributes with a spinning reserve \( SSR(j_k) \) at stage \( k \), the system spinning reserve margin is given by
\[
\Delta = MSSR - SSR(j_k)
\]

(13)

Then, by (3), if \( \Delta \leq 0 \) the constraint is satisfied and re-dispatch is not needed. Conversely, if \( \Delta > 0 \), the constraint is violated and re-dispatch is required.

Assume the condition \( \Delta > 0 \). Then by Lemma 7, we know that we have two sets of units \( I_1 \) and \( I_2 \), as given by Definition 3. Let \( PNG \) be the total generation that can be removed from the \( I_2 \) units to produce an increase in the system spinning reserve, and let \( UPG \) be the total additional generation that can be allocated among the \( I_1 \) units without changing their individual spinning reserves. That is,
\[
UPG = \sum_{i \in I_1} (P_i, \text{MAX} - MSR_i) \cdot p_{ik}
\]

(14)

\[
DNG = \sum_{i \in I_2} (P_i, \text{MAX} - MSR_i)
\]

(15)

We can now state the following theorem

**Theorem.**
If a set \( I_{ON} \) is unfeasible by violating condition (3), it can still be rendered feasible by re-dispatch if and only if conditions (7) and (8) are satisfied.

**Proof:** Provided in Appendix A.

Conditions (7) and (8) actually establish the bounds of the feasibility region in the system spinning reserve \( SSR(j_k) \) versus \( PD_k \) plane, which may be represented graphically as shown in Figure 2. The horizontal line bounding this region corresponds to

\[
SSR( jk ) = SMSR = \sum_{i \in I_{ON}} MSR_i
\]

The sloping line represents the reduction in system spinning reserve attainable by \( I_{ON} \) units as the load \( PD_k \) increased from the breakpoint defined as
\[
PB = \sum_{i \in I_{ON}} P_i, \text{MAX} - \sum_{i \in I_{ON}} MSR_i
\]

The highest possible load that these units can meet is
\[
PH = \sum_{i \in I_{ON}} P_i, \text{MAX} - MSSR
\]

which is also the right side boundary of the feasibility region. The left side boundary is given by
\[
PL = \sum_{i \in I_{ON}} P_i, \text{MIN}
\]

which represents the lowest load that must be delivered.
This feasibility region is subdivided into two mutually exclusive sub-regions A and B. Region A includes the dotted line as its lower bound, while region B excludes it. The units along the upper horizontal boundary are of type $I_1$ whereas along the sloping line, they belong to type $I_2$ units. Excluding these two boundaries, the feasibility region includes both unit types. Also, unit combinations in region A does not require re-dispatch while the converse is true in region B with feasibility of re-dispatch guaranteed.

4. SYSTEM DATA

Although the dynamic programming algorithm can accommodate any number of units, the system chosen for illustration purposes has 20 units. Their characteristics are given in Table 1. The coefficients, or parameters, of the production, startup, and shutdown costs are given in Table 2. The status of the units at the initial stage is given in Table 3. The system load daily profile is specified in Table 4.

### Table 1. Unit Characteristics

<table>
<thead>
<tr>
<th>UNO</th>
<th>FLAG</th>
<th>PNO</th>
<th>PMAX</th>
<th>PMIN</th>
<th>MSR</th>
<th>MU</th>
<th>MD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>443</td>
<td>125</td>
<td>100</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>443</td>
<td>125</td>
<td>100</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>520</td>
<td>250</td>
<td>104</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>520</td>
<td>250</td>
<td>104</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>550</td>
<td>250</td>
<td>110</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>550</td>
<td>250</td>
<td>100</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>4</td>
<td>320</td>
<td>120</td>
<td>70</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4</td>
<td>320</td>
<td>120</td>
<td>70</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>3</td>
<td>280</td>
<td>75</td>
<td>60</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>3</td>
<td>280</td>
<td>75</td>
<td>60</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>3</td>
<td>148</td>
<td>50</td>
<td>90</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2</td>
<td>148</td>
<td>50</td>
<td>90</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>118</td>
<td>25</td>
<td>90</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>2</td>
<td>118</td>
<td>25</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>5</td>
<td>100</td>
<td>30</td>
<td>70</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>4</td>
<td>100</td>
<td>30</td>
<td>70</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>2</td>
<td>80</td>
<td>20</td>
<td>50</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>5</td>
<td>80</td>
<td>20</td>
<td>50</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>1</td>
<td>60</td>
<td>15</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>1</td>
<td>60</td>
<td>15</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Where UNO is the unit number, FLAG=0,1,2, indicates unit not available, unit available, and unit assigned to must run status, respectively, PNO gives the plant number, PMAX and PMIN are the unit maximum and minimum generations in (MW), MSR is the unit maximum spinning reserve, MU and MD are the unit minimum up and minimum down time in (hrs) respectively.

### Table 2. Parameters of Production and Start-Up Cost Functions

<table>
<thead>
<tr>
<th>UNO</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00135</td>
<td>1.1285</td>
<td>100</td>
<td>227</td>
<td>0.20</td>
</tr>
<tr>
<td>2</td>
<td>0.00132</td>
<td>1.1265</td>
<td>110</td>
<td>207</td>
<td>0.20</td>
</tr>
<tr>
<td>3</td>
<td>0.00127</td>
<td>1.1564</td>
<td>105</td>
<td>267</td>
<td>0.10</td>
</tr>
<tr>
<td>4</td>
<td>0.00125</td>
<td>1.1854</td>
<td>115</td>
<td>227</td>
<td>0.10</td>
</tr>
<tr>
<td>5</td>
<td>0.00148</td>
<td>1.2136</td>
<td>82</td>
<td>282</td>
<td>0.10</td>
</tr>
<tr>
<td>6</td>
<td>0.00140</td>
<td>1.1136</td>
<td>92</td>
<td>207</td>
<td>0.10</td>
</tr>
<tr>
<td>7</td>
<td>0.0289</td>
<td>1.2643</td>
<td>92</td>
<td>207</td>
<td>0.10</td>
</tr>
<tr>
<td>8</td>
<td>0.00280</td>
<td>1.2443</td>
<td>69</td>
<td>157</td>
<td>0.20</td>
</tr>
<tr>
<td>9</td>
<td>0.00261</td>
<td>1.5354</td>
<td>52</td>
<td>176</td>
<td>0.10</td>
</tr>
<tr>
<td>10</td>
<td>0.00291</td>
<td>1.7354</td>
<td>52</td>
<td>156</td>
<td>0.10</td>
</tr>
<tr>
<td>11</td>
<td>0.00212</td>
<td>1.8015</td>
<td>29</td>
<td>113</td>
<td>0.10</td>
</tr>
<tr>
<td>12</td>
<td>0.00282</td>
<td>1.7015</td>
<td>59</td>
<td>103</td>
<td>0.10</td>
</tr>
<tr>
<td>13</td>
<td>0.00382</td>
<td>1.6966</td>
<td>32</td>
<td>94</td>
<td>0.10</td>
</tr>
<tr>
<td>14</td>
<td>0.00302</td>
<td>1.6066</td>
<td>42</td>
<td>99</td>
<td>0.10</td>
</tr>
<tr>
<td>15</td>
<td>0.00393</td>
<td>1.8526</td>
<td>40</td>
<td>114</td>
<td>0.10</td>
</tr>
<tr>
<td>16</td>
<td>0.00395</td>
<td>1.8539</td>
<td>40</td>
<td>114</td>
<td>0.10</td>
</tr>
<tr>
<td>17</td>
<td>0.00396</td>
<td>1.9161</td>
<td>40</td>
<td>114</td>
<td>0.10</td>
</tr>
<tr>
<td>18</td>
<td>0.00296</td>
<td>1.8161</td>
<td>25</td>
<td>101</td>
<td>0.10</td>
</tr>
<tr>
<td>19</td>
<td>0.00518</td>
<td>2.3034</td>
<td>25</td>
<td>85</td>
<td>0.30</td>
</tr>
<tr>
<td>20</td>
<td>0.00510</td>
<td>2.2034</td>
<td>15</td>
<td>85</td>
<td>0.30</td>
</tr>
</tbody>
</table>

where $C(P) = aP^2 + bP + d$ [$/hr$], is the production cost function, and $SC(T) = CCS[1-e^{-7TR}] [$/hr$], $T$ is the startup cost function.

### Table 3. Unit Initial Status

<table>
<thead>
<tr>
<th>UNO</th>
<th>PUS</th>
<th>NONH</th>
<th>NOFH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
where \( UNO \) and \( PUS = 0,1 \) indicate unit previously down and previously up, \( NONH \) and \( NOFH \) indicate the number of hours unit has previously been on and off respectively.

Table 4. Load Profile

<table>
<thead>
<tr>
<th>STAGE</th>
<th>MW</th>
<th>STAGE</th>
<th>MW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2000</td>
<td>13</td>
<td>4180</td>
</tr>
<tr>
<td>2</td>
<td>2200</td>
<td>14</td>
<td>4200</td>
</tr>
<tr>
<td>3</td>
<td>2500</td>
<td>15</td>
<td>4205</td>
</tr>
<tr>
<td>4</td>
<td>3000</td>
<td>16</td>
<td>4190</td>
</tr>
<tr>
<td>5</td>
<td>3400</td>
<td>17</td>
<td>4050</td>
</tr>
<tr>
<td>6</td>
<td>3800</td>
<td>18</td>
<td>4010</td>
</tr>
<tr>
<td>7</td>
<td>3890</td>
<td>19</td>
<td>4000</td>
</tr>
<tr>
<td>8</td>
<td>3900</td>
<td>20</td>
<td>3900</td>
</tr>
<tr>
<td>9</td>
<td>4000</td>
<td>21</td>
<td>3800</td>
</tr>
<tr>
<td>10</td>
<td>4050</td>
<td>22</td>
<td>3000</td>
</tr>
<tr>
<td>11</td>
<td>4100</td>
<td>23</td>
<td>2500</td>
</tr>
<tr>
<td>12</td>
<td>4140</td>
<td>24</td>
<td>2100</td>
</tr>
</tbody>
</table>

5. CASE STUDY

As indicated earlier, the computer code is a full dynamic programming implementation of the algorithm that considers the dispatching feasibility of every possible unit combination in the solution space. To illustrate the efficiency achievable by the off-line, or a priori, checking of conditions (7) and (8), four cases were run. In the first, neither of these conditions is checked. In the second, we check condition (7) but not (8), whereas in the third, the reverse is true. Finally, the last test includes the checking of both conditions. The results are presented in Table 5 where, for each case, we provide the number of dispatches attempted, the number of dispatches that passed by meeting conditions (1) and (2) but not necessarily condition (3), and the actual number of re-dispatches performed, and finally, the number of re-dispatches passed. We also include the CPU time required for each case, which is 30% faster when both conditions were checked as indicated.

The solution for all four cases has the same optimal operation cost and thus yielding the same optimal schedule as shown in Table VI. When neither condition (7) nor (8) are tested, as in the first case, the algorithm requires 6.5 additional CPU minutes to deliver the solution compared to the last case, where both conditions were checked a priori yielding higher efficiency.

We note that for the last case, where both conditions (7) and (8) are checked, the number of redispatches attempted and passed is identical. This is expected since these conditions are both necessary and sufficient. When only one of the two conditions is tested off-line, as in the second and third cases, the solution speed increases somewhat relative to the first case, but do not differ greatly whether one condition or the other is checked. However, they are included here for completeness.

Table 5. Computer Results

<table>
<thead>
<tr>
<th>Conditions</th>
<th>ED Numbers</th>
<th>ED Passed</th>
<th>RED Numbers</th>
<th>RED Passed</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE CHECKED</td>
<td>98,304</td>
<td>38,238</td>
<td>26,599</td>
<td>6,322</td>
<td>20:54</td>
</tr>
<tr>
<td>(7) BUT NOT (8)</td>
<td>82,104</td>
<td>34,843</td>
<td>23,204</td>
<td>6,322</td>
<td>16:49</td>
</tr>
<tr>
<td>(8) BUT NOT (7)</td>
<td>18,783</td>
<td>18,783</td>
<td>7,144</td>
<td>6,322</td>
<td>16:17</td>
</tr>
<tr>
<td>BOTH CHECKED</td>
<td>17,961</td>
<td>17,961</td>
<td>6,322</td>
<td>6,322</td>
<td>14:23</td>
</tr>
</tbody>
</table>

where \( ED = \) Economic Dispatch, \( RED = \) Economic Re-dispatch. *Necessary and Sufficient Feasibility Conditions.

5. CASE STUDY

As indicated earlier, the computer code is a full dynamic programming implementation of the algorithm that considers the dispatching feasibility of every possible unit combination in the solution space. To illustrate the efficiency achievable by the off-line, or a priori, checking of conditions (7) and (8), four cases were run. In the first, neither of these conditions is checked. In the second, we check condition (7) but not (8), whereas in the third, the reverse is true. Finally, the last test includes the checking of both conditions. The results are presented in Table 5 where, for each case, we provide the number of dispatches attempted, the number of dispatches that passed by meeting conditions (1) and (2) but not necessarily condition (3), and the actual number of re-dispatches performed, and finally, the number of re-dispatches passed. We also include the CPU time required for each case, which is 30% faster when both conditions were checked as indicated.

The solution for all four cases has the same optimal operation cost and thus yielding the same optimal schedule as shown in table VI. When neither condition (7) nor (8) are tested, as in the first case, the algorithm requires 6.5 additional CPU minutes to deliver the solution compared to the last case, where both conditions were checked a priori yielding higher efficiency.

We note that for the last case, where both conditions (7) and (8) are checked, the number of redispatches attempted and passed is identical. This is expected since these conditions are both necessary and sufficient. When only one of the two conditions is tested off-line, as in the second and third cases, the solution speed increases somewhat relative to the first case, but do not differ greatly whether one condition or the other is checked. However, they are included here for completeness.

Table 6. Optimal Schedule

<table>
<thead>
<tr>
<th>K</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

where: \( 1 = \) Unit on-line, \( 0 = \) Unit off-line

Y. Al Kalaani
6. CONCLUSIONS
In this paper, we have derived the mathematical formulation for the unit commitment problem in power systems using dynamic programming techniques. Two conditions were presented, which can be checked offline to eliminate unit combinations that are guaranteed a priori to be infeasible combinations since they are both necessary for a successful solution. Furthermore, we have presented a theorem with necessary and sufficient conditions for unit feasibility that require re-dispatch to meet the system spinning reserve requirement. Detailed examples on a 20-unit system showed specifically how the computational efficiency was attained. Therefore, when these conditions are checked as prescribed, they will always render very efficient solutions to unit commitment algorithms.

REFERENCES

APPENDIX A

Proof of Lemma 5
From (1) and (2) it immediately follows that

\[ PD_k \geq \sum_{i \in I^O} P_{i, \text{MAX}} \]

which is the left hand side of (8). Now consider a dispatch for which \( I_{ON} \) as given in Definition 3, then (4) can be written as

\[ \text{SSR}(k) = \sum_{i \in I^1} MSR_i + \sum_{i \in I^2} P_{i, \text{MAX}} \sum_{i \in I^2} P_i k \geq MSSR \] (9)

Rewriting (1) as

\[ \sum_{i \in I_2} P_{i,k} = PD_k - \sum_{i \in I_2} P_{i,k} \] (10)

and substituting (10) into (9) yields

\[ \sum_{i \in I_1} (MSR_i + P_{i,k}) + \sum_{i \in I_2} P_{i, \text{MAX}} - PD_k \geq MSSR \] (11)

From (5) we can write

\[ P_{i, \text{MAX}} \geq MSR_i + P_{i, \text{MAX}} \quad i \in I_1 \]

and substituting into (10) gives

\[ \sum_{i \in I_1} P_{i, \text{MAX}} + \sum_{i \in I_2} P_{i, \text{MAX}} \geq MSSR + PD_k \] (12)

and, by rearranging and combining the two sets of units, the right hand side of (8) follows.

APPENDIX B

Proof of the theorem
a) Necessity: Consider a re-dispatch for which (3) is met. Then the set \( I_{ON} \) is rendered feasible according to Definition 2 and, by Lemmas 4 and 5, both (7) and (8) are satisfied.

b) Sufficiency: Assume that (7) and (8) are satisfied. Rewriting (14) as

\[ \sum_{i \in I_1} MSR_i = \sum_{i \in I_1} P_{i, \text{MAX}} - \sum_{i \in I_1} P_i k - UPG \]

and (15) as

\[ \sum_{i \in I_2} P_{i, \text{MAX}} - \sum_{i \in I_2} P_i k = \sum_{i \in I_2} MSR_i - DNG \]

Using (4), (16), and Lemma 7, we may write (13) as

\[ \Delta = MSSR - \sum_{i \in I_1} P_{i, \text{MAX}} + \sum_{i \in I_1} P_i k + UPG \\
- \sum_{i \in I_2} P_{i, \text{MAX}} - \sum_{i \in I_2} P_i k \]

Using the union of the two sets of units and (1), we can solve for \( UPG \) to yield

\[ UPG = \Delta + \sum_{i \in I_{ON}} P_{i, \text{MAX}} - MSSR - PD_k \]

Since \( \Delta > 0 \) and by hypothesis, (8) is satisfied, then \( UPG \geq \Delta \).

Similarly, from (4), (17), and Lemma 7 we can write (13) as

\[ \Delta = MSSR - \sum_{i \in I_1} MSR_i - \sum_{i \in I_2} MSR_i + DNG \]

Again, using the union of the two sets, we can solve for \( DNG \) yielding

\[ DNG = \Delta + \sum_{i \in I_{ON}} MSR_i - MSSR \]

Since \( \Delta > 0 \) and by hypothesis, (7) is satisfied, then \( DNG \geq \Delta \). In [6] it was shown that \( UPG \geq \Delta \) and \( DNG \geq \Delta \) are both necessary and sufficient for the feasibility of re-dispatch. Therefore, sufficiency is proven.