CHAPTER 4

REINFORCEMENT LEARNING APPROACHES
FOR SOLUTION OF
UNIT COMMITMENT PROBLEM

4.1 Introduction

Unit Commitment Problem (UCP) in power system refers to the problem of determining the on/off status of generating units that minimize the operating cost during a given time horizon. Formulation of exact mathematical model for the same is difficult in practical situations. Cost associated with the different generating units is also different and random. Most often it is difficult to obtain a precise cost function for solving UCP. Also availability of the different generating units is different during each time slot due to the numerous operating constraints.

The time period considered for this short term scheduling task varies from 24 hours to one week. Due to the large number of ON/OFF combinations possible, even for small number of generating units and short period of time, UCP is a complex optimization problem. Unit Commitment has been formulated as a non linear, large-scale, mixed integer combinational optimization problem (Wood and Wollenberg [2002]).

From the review of the existing strategies, mainly two points can be concluded:

(i) Conventional methods like Lagrange Relaxation, Dynamic Programming etc. find limitation for higher order problems.

(ii) Stochastic methods like Genetic Algorithm, Evolutionary Programming etc. have limited computational efficiency when a large number of units involved.
Since Reinforcement Learning has been found to be a good tool for many of
the optimization problems, it appeared to be very much promising to solve this
scheduling problem using Reinforcement Learning.

In this chapter, a stochastic solution strategy based on Reinforcement Learning
is proposed. The class of algorithms is termed as RL_UCP. Two varieties of
exploration methods are used: ε greedy and pursuit method. The power generation
constraints of the units, minimum up time and minimum down time are also considered
in the formulation of RL solution. A number of case studies are made to illustrate the
reliability and flexibility of the algorithms.

In the next section, a mathematical formulation of the Unit Commitment
Problem is given. For developing a Reinforcement Learning solution to Unit
Commitment problem, it is formulated as a multi stage decision making task. The
Reinforcement solution to simple Unit Commitment problem is reviewed (RL_UCP1).
An efficient solution using pursuit method without considering minimum up time and
minimum down time constraints is suggested (RL_UCP2). Then the minimum up time
and down time constraints are incorporated and a third algorithm (RL_UCP3) is
developed to solve the same. To make the solution more efficient one, an algorithm
with state aggregation strategy is developed (RL_UCP4).

4.2 Problem Formulation

Unit Commitment Problem is to decide which of the available units has to be
turned on for the next period of time. The decision is subject to the minimization of
fuel cost and to the various system and unit constraints. At the system level, the
forecasted load demand should be satisfied by the units in service. In an interconnected
system, the load demand should also include the interchange power required due to the
contractual obligation between the different connected areas. Spinning reserve is the
other system requirement to be satisfied while selecting the generating units. In
addition, individual units are likely to have status restrictions during any given time
period. The problem becomes more complicated when minimum up time and down
time requirements are considered, since they couple commitment decisions of successive hours.

The main objective of this optimization task is to minimize the total operating cost over the scheduled time horizon, while satisfying the different operational constraints. The operating cost includes start up cost, shut down cost, running cost, maintenance cost etc. The UCP can be formulated as:

Minimize Operational cost

Subject to

- Generation constraints
- Reserve constraints
- Unit capacity limits
- Minimum Up time constraints
- Minimum Down time constraints
- Ramp rate constraints
- Unit status restrictions

4.2.1 Objective

As explained above, the objective of UCP is the minimization of total operating cost over the complete scheduling horizon. The major component of the operating cost for thermal units is the fuel cost. This is represented by an input / output characteristics which is normally approximated as polynomial curve (quadratic or higher order) or as a piecewise linear curve. For quadratic cost, the cost function is of the form

\[ C_i(P_{i,k}) = a_i + b_i P_{i,k} + c_i P_{i,k}^2, \]

where \( a_i, b_i \) and \( c_i \) are cost coefficients,

\[ P_{i,k} \] - Power generated by \( i^{th} \) unit during hour \( k \)

If a unit is down for a time slot, it can be brought back to operation by incurring an extra cost, which is due to the fuel wastage, additional feed water and energy needed
for heating. Accordingly, the total fuel cost $F_T$ which is the objective function of UCP is:

$$F_T = \sum_{k=1}^{T} \sum_{i=1}^{N} [C_i (P_{i,k})u_{i,k} + ST_i (u_{i,k})(1 - u_{i,k-1})]$$

where $T$ is the time period (number of hours) considered, $C_i(P_{i,k})$ is the cost of generating power $P_i$ during $k^{th}$ hour by $i^{th}$ unit, $ST_i$ is the start up cost of the $i^{th}$ unit, $u_{i,k}$ is the status of the $i^{th}$ unit during $k^{th}$ hour and $u_{i,k-1}$ is the status of the $i^{th}$ unit during the previous hour.

4.2.2. The constraints

The variety of constraints to UCP can be broadly classified as System constraints and Unit constraints.

**System Constraints:**

- **Load demand constraint:** The generated power from all the committed or on line units must satisfy the load balance equation

$$\sum_{i=1}^{N} P_{i,k}u_{i,k} = l_k; \quad 1 \leq k \leq T,$$

where $l_k$ is the load demand at hour $k$.

**Unit Constraints:**

- **Generation capacity constraints:** Each generating unit is having the minimum and maximum capacity limit due to the different operational restriction on the associated boiler and other accessories

$$P_{\text{min}(i)} \leq P_{i,k} \leq P_{\text{max}(i)}; \quad 0 \leq i \leq N - 1, \quad 1 \leq k \leq T$$

- **Minimum up time and down time constraint:** Minimum up time is the number of hours unit $i$ must be ON before it can be turned OFF. Similarly, minimum down time restrict it to turn ON, when it is DOWN. If $t_{off}i$ represents the number of hours $i^{th}$ unit has been shut down, $t_{on}i$ the number of hours $i^{th}$ unit has been on line, $U_i$ the minimum up time and $D_i$ the minimum down time corresponding to $i^{th}$ unit, then these constraints can be expressed as:

$$t_{off}i \geq D_i; \quad t_{on}i \geq U_i, \quad 0 \leq i \leq N - 1$$
Ramp rate limits: The ramp rate limits restrict the amount of change of generation of a unit between two successive hours.

\[ P_{i,k} - P_{i,(k-1)} \leq UR_i \]

\[ P_{i,(k-1)} - P_{i,k} \leq DR_i \]

where \( UR_i \) and \( DR_i \) are the ramp up and ramp down rates of unit \( i \).

Unit status restrictions: Some of the units will be given the status of 'Must Run' or 'Not available' due to the restrictions on the availability of fuel, maintenance schedule etc.

4.3 Mathematical model of the problem

The mathematical description of the problem considered can be summarized as:

Minimize the objective function,

\[
F_T = \sum_{k=1}^{T} \sum_{i=1}^{N} \left[ C_i (P_{i,k}) u_{i,k} + S i (u_{i,k}) (1 - u_{i,k-1}) \right]
\]

subject to the constraints,

\[
\sum_{i=1}^{N} P_{i,k} u_{i,k} = l_k; \quad 1 \leq k \leq T
\]

(4.2)

\[
P_{\text{min} (i)} \leq P_{i,k} \leq P_{\text{max} (i)}, \quad 0 \leq i \leq N-1, \quad 1 \leq k \leq T
\]

(4.3)

\[
t_{\text{off} (i)} \geq D_i; \quad t_{\text{on} (i)} \geq U_i, \quad 0 \leq i \leq N-1
\]

(4.4)

In order to formulate a Reinforcement Learning approach, in the next section UCP is formulated as a multi stage decision task.
4.4 Unit Commitment as a Multi Stage decision making task

Consider a Power system having \( N \) generating units intended to meet the load profile forecasted for \( T \) hours, \((l_0, l_1, l_2, \ldots, l_T)\). The Unit Commitment Problem is to find which all units are to be committed in each of the slots of time. Objective is to select units so as to minimize the cost of generation, at the same time meeting the load demand and satisfying the constraints. That is to find a decision or commitment schedule \( a_0, a_1, a_2, \ldots, a_T \), where \( a_k \) is a vector representing the status of the \( N \) generating units during \( k^{th} \) hour.

\[
a_k = [a_k^0, a_k^1, \ldots, a_k^{N-1}]
\]

\( a_k^i = 0 \) indicates the OFF status of \( i^{th} \) unit during \( k^{th} \) time slot while \( a_k^i = 1 \) indicates the ON status.

For finding the schedule of \( T \) hour load forecast, it can be modeled as a \( T \) stage problem. While defining an MDP or Reinforcement Learning problem, state, action, transition function and reward function are to be identified with respect to the scheduling problem.

In the case of Unit Commitment Problem, the state of the system at any time slot (hour) \( k \) can represent the status of each of the \( N \) units. That is, the state \( x_k \) can be represented as a tuple \( (k, p_k) \) where \( p_k \) is a string of integers, \([p_k^0, p_k^1, \ldots, p_k^{N-1}]\), \( p_k^i \) being the integer representing the status of \( i^{th} \) unit. When the minimum up time and down time constraints are neglected the ON / OFF status of each unit can be used to represent \( p_k^i \). Then the integer \( p_k^i \) will be binary; ON status represented by \('1'\) and OFF status by \('0'\). Consideration of the minimum up time and minimum down time constraints force to include the number of time units each unit has been ON /OFF in the state representation. Then the variable \( p_k^i \) can take positive or negative value ranging from \(-D_i\) to \(+U_i\).

The part of the state space at time slot or stage \( k \) can be denoted by

\[
\chi_k = \{(k, [p_k^0, p_k^1, \ldots, p_k^{N-1}])\}
\]

and the entire state space can be defined as

\[
\chi = \chi_0 \cup \chi_1 \cup \ldots \chi_{T-1}
\]
Next is to identify the actions or decisions at each stage of the multi stage problem. In case of UCP, the action or decision on each unit is either to commit or not the particular unit during that particular hour or time slot. Therefore action set at each stage $k$ can be defined as $\mathcal{A}_k = \{ [a^0_k, a^1_k, \ldots, a^{N-1}_k], a^i_k = 0 \text{ or } 1 \}$. When certain generating units are committed during particular hour $k$, i.e, $a^i_k = 1$ for certain values of $i$, then the load demand or power to be generated by these committed units is to be decided. This is done through an Economic Dispatch solution.

The next part to be defined in this MDP is the transition function. Transition function defines the transition from the current state to the next state on applying an action. That is, from the current state $x_k$, taking an action $a_k$, it reaches the next state $x_{k+1}$. Since the action is to make the units ON/OFF, the next state $x_{k+1}$ is decided by the present state $x_k$ and action $a_k$. Transition function $f(x_k, a_k)$ depends on the state representation.

Last part to be defined is the reinforcement function. It should reflect the objectives of the Unit Commitment Problem. Unit Commitment Problem can have multiple objectives like minimization of cost, minimizing emissions from the thermal plants etc. Here, the minimization of total cost of production is taken as the objective of the problem. The total reward for the $T$ stages should be the total cost of production. Therefore, the reinforcement function at $k^{th}$ stage is defined as the cost of production of the required amount of generation during the $k^{th}$ period.

That is,

$$g(x_k, a_k, x_{k+1}) = \sum_{i=0}^{N-1} [C_i(P_{i,k})u_{i,k} + ST_{i}(u_{i,k})(1 - u_{i,k-1})]$$

Here, $P_{i,k}$ is the power generation by $i^{th}$ unit during $k^{th}$ time slot and $u_{i,k}$ is the status of $i^{th}$ unit during $k^{th}$ time slot.

In short, Unit Commitment Problem is now formulated as a Multi Stage decision making problem, which passes through $T$ stages. At each stage $k$, from one of the states $x_k = (k, p_k)$ an action or allocation $a_k$ is chosen depending on some exploration strategy. Then a state transition occurs to $x_{k+1}$ based on the transition
function. Each state transition results in a reward corresponding to power allocation to the committed units. Then the problem reduces to finding the optimum action $a_k$ at each state $x_t$ and corresponding to each time slot $k$.

In the next sections a class of Reinforcement Learning solutions is proposed. In all these algorithms the action space and the reinforcement function are the same. The definition of state space, transition function and the update strategy are different.

4.5 Reinforcement Learning Approach to Unit Commitment Problem

Having formulated as a Multistage Decision Problem, implementable solutions are developed using Reinforcement Learning approach. First a review of the basic algorithm is given. Neglecting the minimum up time and minimum down time constraints and using exploration through $\varepsilon$ - greedy strategy, solution is presented (RL_UCP1). Then employing pursuit method for action selection, algorithm for solution is proposed (RL_UCP2).

Next, Minimum up time and down time constraints are incorporated which needs the state of the system (status of the units) to be represented as integer instead of binary representation in the previous solutions. To handle the large state space, an indexing method is proposed while developing solution (RL_UCP3). A more efficient solution is then proposed using state aggregation strategy. In the next sections, the solution methods and algorithms are presented in detail.

4.6 $\varepsilon$ - greedy algorithm for Unit Commitment Problem

neglecting minimum up time and minimum down time

(RL_UCP1)

In the previous section, a $T$ hour allocation problem is modeled as a $T$ stage problem. In this section, an algorithm based on Reinforcement Learning and using $\varepsilon$- greedy exploration strategy is presented. For the same, different parts of Reinforcement Learning solution are first defined precisely.

When the minimum up time and minimum down time are neglected, the state of the system at any hour $k$ can be represented by the ON /OFF status of the units.
Hence, state can be represented by the tuple, \( x_k = (k, p_k) \), where \( p_k = [p_k^0, p_k^1, \ldots, p_k^{N-1}] \) and \( p_k^i = 1 \) if the \( i^{th} \) unit is ON, \( p_k^i = 0 \) if the \( i^{th} \) unit is OFF.

The part of the state space at time slot or stage \( k \) can be denoted by

\[
\chi_k = \{(k, [p_k^0, p_k^1, \ldots, p_k^{N-1}]), p_k^i = 0 \text{ or } 1\}
\]

The state space can be defined as,

\[
\chi = \chi_0 \cup \chi_1 \cup \ldots \chi_{T-1}
\]

Action or decision at any stage of the problem is the decision of making ON / OFF of a unit. Therefore the action set at stage \( k \) is represented by,

\[
\mathcal{A}_k = \{[a_k^0, a_k^1, \ldots, a_k^{N-1}], a_k^i = 0 \text{ or } 1\}.
\]

The transition function defines the change of state from \( x_k \) to \( x_{k+1} \). In this case, the next state (in RL terminology) is just the status of units after the present action or decision. Therefore the transition function \( f(x_k, a_k) \) is defined as,

\[
x_{k+1} = a_k
\]

Lastly, the reward is the cost of allocating the committed units with power \( P_{ik} \) \( i = 0, \ldots, N-1 \) and status of the unit \( u_{i,k}=1 \). Thus the reward function,

\[
g(x_k, a_k, x_{k+1}) = \sum_{i=0}^{N-1} [C_i(P_{ik})u_{ik} + ST_i(u_{ik})(1 - u_{i,k-1})]
\]

(4.5)

For easiness of representation, the binary string in the state as well as action can be represented by the equivalent decimal value. The state at any stage can be represented as a tuple \((k, d)\) where \( k \) represents the stage or time slot and \( d \) represents the decimal equivalent of the binary string representing the status. For example \((2, 4)\) represent the state \((2, [0100])\) which indicate the status 0100 during 2\(^{nd}\) hour for a four unit system. Or in other words it indicates only unit 1 is ON during 2\(^{nd}\) hour.

Now a straight forward solution using Q learning is suggested for solving this MDP. Estimated Q values of each state – action pairs are stored in a look up table as \( Q(x_k, a_k), x_k \) having the information on the time slot and present status of the different
units. At each step of the learning phase, the algorithm updates the Q value of the corresponding state – action pair. The algorithm (RL_UCP1) for the learning phase is described below:

The initial status of the different generating units is read from the unit data. Then the different possible states and actions possible are identified. Q value corresponding to different state – action pairs are initialized to zero.

The generating units are having their minimum and maximum generation limits. At each slot of time, the unit combinations or actions should be in such a way as to satisfy the load requirement. Therefore, using the forecasted load profile and the unit generation constraints, the set of feasible actions $A_k$ is identified for each stage $k$ of the multi stage decision problem. Using the $\varepsilon$-greedy strategy one of the actions $a_k$ from the permissible action set $A_k$ is selected. Depending on the action selected, state transition occurs to next stage $k+1$ as $x_{k+1} = a_k$. The reward of state transition or action is calculated using the power allocation to each unit $P_{ik}$ through dispatch algorithm and using the equation (4.5).

The cost function can be non convex in nature and can be represented either in piece wise quadratic form or higher order polynomial form. While finding the dispatch among the committed units, for simplicity, linear cost function is taken. This method gives only tolerable error. After that, cost of generation is obtained using the given cost functions. In this approach of solution a defined mathematical cost function is not at all necessary. The net cost of generation is taken as the reward $g(x_k, a_k, x_{k+1})$. Using the reward, estimated Q value of the corresponding state – action pair is updated at each of the stages until the last stage using the equation:

$$Q^{n+1}(x_k, a_k) = Q^n(x_k, a_k) + \alpha [g(x_k, a_k, x_{k+1}) + \gamma \min_{a' \in A_{k+1}} Q^n(x_{k+1}, a') - Q^n(x_k, a_k)]$$

(4.6)

here, $\alpha$ is the step size of learning and $\gamma$ is the discount factor.
During the last stage \( k = T \), since there is no more future stages the second term in the update equation will turn to be zero, the updating is carried out as

\[
Q^{n+1}(x_k, a_k) = Q^n(x_k, a_k) + \alpha \left[ g(x_k, a_k, x_{k+1}) - Q^n(x_k, a_k) \right]
\]

(4.7)

This constitutes one episode. In each episode the algorithm passes through all the \( T \) stages. Then the algorithm is executed again from the initial state \( x_p \). These episodes are repeated a large number of times. If \( \alpha \) is sufficiently small and if all possible \((x_k, a_k)\) combinations of state and action occur sufficiently often then the above iteration will result in \( Q^* \) converging to \( Q^* \) (Bertsekas and Tsitsikilis [1996], Sutton and Barto [1998]).

In the initial phases of learning the estimated \( Q \) values, \( Q^n(x_k, a_k) \) may not be closer to the optimum value \( Q^*(x_k, a_k) \). As the learning proceeds, the estimated \( Q \) values turn to be better. When the estimated \( Q \) values approach to optimum, change in the value in two successive iterations will be negligibly small. In other words, \( Q^{n+1}(x_k, a_k) \) will be the same as \( Q^n(x_k, a_k) \).

In order to apply the proposed Reinforcement Learning algorithms, first suitable values of the learning parameters are to be selected. Value of \( \varepsilon \) balances the rate of exploration and exploitation. A small fixed value result in premature convergence, while a large fixed value may make the system oscillatory. For balancing exploration and exploitation, a reasonable value between 0 and 1 is taken for the learning parameter \( \varepsilon \) initially and is decreased by a small factor successively.

In the learning procedure, a block of consecutive iterations are examined for modification in the estimated \( Q \) values. If the change is negligibly small in all these iterations, the estimated \( Q \) values are regarded as optimum corresponding to a particular state – action pair. The iteration number thus obtained can be taken as
maximum number of iterations in the learning algorithm. The learning steps are described as RL_UCP1.

*Algorithm for Unit Commitment solution using \( \varepsilon \)-greedy (RL_UCP1)*

1. Read the unit data
2. Read the initial status \( x_0 \)
3. Read the forecast for the next \( T \) hours
4. Identify the feasible states and actions
5. Initialize \( Q^0(x, a) = 0 \) \( \forall x \in X, \forall a \in A \)
6. Initialize \( k = 0 \)
7. Initialize \( \varepsilon = 0.5, \alpha = 0.1 \) and \( \gamma = 1 \)
8. For \( n = 0 \) to max _iteration
   9. Begin
      10. For \( (k = 0 \) to \( T-1) \)
          11. Do
              12. Choose an action using \( \varepsilon \)-greedy algorithm
              13. Find the next state \( x_{k+1} \)
              14. Calculate the reward using equation (4.5)
              15. If \( (k < T-1) \) Update the \( Q^n \) to \( Q^{n+1} \) using equation (4.6)
                  Else update \( Q^n \) to \( Q^{n+1} \) using equation (4.7)
          16. End do
      17. Update the value of \( \varepsilon \)
   18. End
19. Save \( Q \) values.
4.7 Pursuit algorithm for Unit Commitment without considering minimum up time and minimum down time (RL_UCP2)

As explained before, in case of pursuit algorithm, actions are selected based on a probability distribution function \( p_{x_k}(\cdot) \). This probability distribution function is updated as the algorithm proceeds.

In the solution of Unit Commitment problem, initially the probability associated with each action \( a_k \) in the action set \( A_k \) corresponding to \( x_k \) are initialized with equal values as

\[
p_{x_k}(a_k) = \frac{1}{n_k}.
\]

\( n_k \) - Total number of permissible actions in state \( x_k \).

As in the previous algorithm, Q values of all state – action pairs are initialized to zero. Then at each iteration step, an action \( a_k \) is selected based on the probability distribution. On performing action \( a_k \), state transition occurs as \( x_{k+1} = a_k \).

The cost incurred in each step of learning is calculated as the sum of cost of producing power \( l_k \) with the generating units given by the binary string ‘s’ in \( a_k \) and the cost associated with ‘s’ as given in the equation (4.5). Q values are then updated using the equation (4.6). At each of the iteration of learning, the greedy action as \( a_g = \arg\min_{a \in A_k} Q(x_k, a) \) is found. Then the corresponding probabilities of actions in the action set are also updated as:

\[
\begin{align*}
    p_{x_k}^{n+1}(a_k) &= p_{x_k}^n(a_k) + \beta [1 - p_{x_k}^n(a_k)], \text{when } a_k = a_g \\
    p_{x_k}^{n+1}(a_k) &= p_{x_k}^n(a_k) - \beta [p_{x_k}^n(a_k)], \text{when } a_k \neq a_g
\end{align*}
\]

(4.8)
The algorithm proceeds through several iterations when ultimately the probability of best action in each hour is increased sufficiently which indicate convergence of the algorithm. As in the previous solution, learning steps can be stopped when the change of Q values in a set of successive iterations are tolerably small, which gives the maximum number of iterations required for the learning procedure. The entire algorithm is given in RL_UCP2:

**Algorithm for Unit Commitment using Pursuit method (RL_UCP2)**

read the Unit data and Load data for T hours

find out set of possible states (χ) and actions (A)

- read the learning parameters
- read the initial status of units $x_0$
- initialise $Q(x,a) = 0$, $\forall x \in \chi$ and $\forall a \in A$
- identify the feasible action set in each hour $k$ as $A_k$
- initialize $p^0_{x_k}(a_k)$ to $1/n_k$, where n_k is the number of actions in $A_k$

for $n = 0$ to max _iteration

begin

for $k = 0$ to $T-1$

do

choose action $a_k$ based on the current probability distribution $p_{x_k}(\cdot)$

find the next state $x_{k+1}$

calculate $g(x_k, a_k, x_{k+1})$

update $Q^n$ to $Q^{n+1}$ using equation (4.6) and (4.7)

update probability $p^n_{x_k}(a_k)$ to $p^{n+1}_{x_k}(a_k)$ using equation (4.8)

end do

end

save $Q$ values.
Since the exploration is based on probability distribution and the probability is updated each and every time an action is selected, the speed of reaching the optimum is improved by this strategy. In the simulation section the performance of this algorithm is compared with the \( \varepsilon \)-greedy method using several standard systems.

### 4.8 Policy Retrieval

Once the learning phase is completed, the schedule of the generating units corresponding to the given load profile can be retrieved. During the learning phase Q values of the state–action pairs will be modified and will approach to optimum. Once the optimum Q values are reached, the best action will be the greedy action at each stage \( k \).

\[
a_k^* = \arg \min_{a_k \in A_k} \{ Q(x_k, a_k) \}, k = 0, \ldots, T - 1.
\]  

(4.9)

Algorithmic steps for finding the optimum schedule \( [a_0^*, a_1^*, \ldots, a_{T-1}^*] \) are detailed below:

**Policy Retrieval steps:**

1. **Read the Q values**
2. **Get the initial status of the units, \( x_0 \)**
3. **For** \( (k = 0 \text{ to } T-1) \)
4. **Do**
   1. **Find the greedy action** \( a_k^* \) **using equation (4.9)**
   2. **Find the next state,** \( x_{k+1} = a_k \)
5. **End do.**

For the above two algorithms, the unit and system constraints are considered except the minimum up time and minimum down time. Now, to incorporate the minimum up time and minimum down time, algorithms are extended in the next sections.
4.9 Reinforcement Learning algorithm for UCP, considering minimum up time and minimum down time (RL_UCP3)

In case of Unit Commitment Problem, one important constraint comes from the minimum up time and minimum down time limitation of the units. Therefore the ‘state’ of the system should essentially indicate the number of hours the unit has been UP or DOWN. Then only the decision to turn on or turn off will be valid. Therefore the state representation used in the previous sections cannot be used further.

For resolving this issue, the state representation is modified. Status of each unit is represented by a positive or negative number indicating the number of hours it has been ON or OFF. Thus, at each stage or hour, system state will be of the form, \( x_k = (k, p_k) \), where \( p_k = [p_k^0, p_k^1, \ldots, p_k^{N-1}] \). Each \( p_k^i \) has positive or negative value corresponding to the number of hours the unit has been UP or DOWN. For example, if the state of a four generating unit system during 2\(^{nd}\) hour is given as \( x_2 = (2, [-2, 1, 2, -1]) \), it gives the information that first and fourth units have been DOWN for two hours and one hour respectively and second and third units have been UP for one hour and two hours respectively.

In principle, for a \( T \) stage problem \( p_k^i \) can take any value between \(-T\) to \( +T\). That is, \( \chi_k = \{(k, [p_k^0, p_k^1, \ldots, p_k^{N-1}]) \mid p_k^i \in \{-T, -T+1, -T+2, \ldots, +T\}\} \).

For such a choice state space will be huge. It may be mentioned here that \( x_k \) is the state of the system as viewed by the learning agent and it need not contain all the information regarding the system. Or in other words, only sufficient information need to be included in the state. For example, in the previous formulations it does not matter how many hours the unit was on, what matters to the learner is whether the unit is ON or OFF. Hence in that case \( p_k^i \in \{0, 1\} \).

Here, when considering the minimum up time and down time, it is immaterial whether the unit has been ON for \( U_i \) hours or \( U_i + L \) hours (where \( U_i \) is the minimum
up time). Therefore, if a unit is ON for more than $U_i$ hours $p_k^{i}$ is taken as $U_i$. Similar is the case with $D_i$. Hence, $p_k^{i} \in \{-D_i, -D_i + 1, \ldots, U_i\}$.

Thus the sub space corresponding to stage $k$,

$$x_k = \{k, [p_k^0, p_k^1, \ldots, p_k^{N-1}] | p_k^{i} \in \{-D_i, -D_i + 1, \ldots, U_i\}\}$$

The number of elements in $\{-D_i, -D_i + 1, \ldots, U_i\}$ is $D_i + U_i$.

Therefore, the number of elements in $\chi = T (D_0 + U_0) (D_1 + U_1) \ldots \ldots (D_{N-1} + U_{N-1})$.

For a six generating unit system with minimum up time and down time of 5 hours for each of the units, the number of states in the state space will be $10^6 \times T$ which is a large number. Therefore storing Q-values for all the possible state action pairs is a cumbersome task. To resolve the same a novel method is proposed in the next section.

Regarding the action space, as in the previous solution, each action represents the ON/OFF status of the units. For an $N$ generating unit system due to the different combinations of ON-OFF status, there will be $2^{N+1}$ actions possible. At each stage depending on the generation constraints enforced by the generating units and the load demand to be met there exists a permissible set of actions,

$$A_k = \{[a_k^0, a_k^1, \ldots, a_k^{N-1}] | a_k^{i} = 0 \text{ or } 1 \}$$

For making the new algorithm simpler, an action is selected based on $\epsilon$-greedy exploration strategy. Each action selection is accompanied by a state transition. In this case, it should account the number of hours one unit has been UP or DOWN.

Therefore, the transition function is to transform the state $x_k = (k, [p_k^0, p_k^1, \ldots, p_k^{N-1}])$ to $x_{k+1} = (k + 1, [p_{k+1}^0, p_{k+1}^1, \ldots, p_{k+1}^{N-1}])$ and is defined as:
\[ p_{k+1} = p_k + 1, \quad \text{if } p_k \text{ positive}, a_k = 1 \]
\[ p_{k+1} = -1, \quad \text{if } p_k \text{ positive}, a_k = 0 \]
\[ p_{k+1} = p_k - 1, \quad \text{if } p_k \text{ negative}, a_k = 0 \]
\[ p_{k+1} = +1, \quad \text{if } p_k \text{ negative}, a_k = 1 \]
\[ p_{k+1} = U_i, \quad \text{if } p_k > U_i \]
\[ p_{k+1} = D_i, \quad \text{if } p_k < D_i \]

(4.10)

Since each action corresponds to a particular unit combination of generating units to be on-line, the cost of generation or reward will be the function \( g(x_k, a_k, x_{k+1}) \) as given in equation (4.5).

Q learning described previously is used for solution of this MDP. Q values of each state – action pairs are to be stored to find the good action at a particular state. In this Unit Commitment Problem, when the minimum up time and down time constraints are taken, the possible states come from a very large state space. Straight forward method of storing the \( Q(x_k, a_k) \) values is using a look up table. But all the states in this huge state space are not relevant. Therefore a novel method of storing Q values is suggested.

Q values of only those states which are encountered at least once in the course of learning are stored. Since the learning process allows sufficient large number of iterations, this seems to be a valid direction. The states are represented by an index number \( (\text{ind } x_i) \), which is an integer and initialized at the first time of encountering the state. For example the tuple, \( (5, (2, [-2, 1, 1, 2]) \) denote the state \( (2, [-2, 1, 1, 2]) \) with an index number '5'.

Similarly, the actions in the action space can also be represented by an integer, which is the decimal equivalent of the binary string representing the status of the different units. The index value of action 0011 is '3'. Using these indices for the state and action strings, the Q values of the different state action pairs can be stored very easily. Q(5, 3) indicate the Q value corresponding to the state \( (2, [-2, 1, 1, 2]) \), which is having index value 5 and action [0 0 1 1].
The possible number of states $n_{states}$ is initialized depending on the number of units $N$ and the minimum down time and minimum up time of the units, since it depends on number of combinations possible with $N$ units as well as the given values of minimum up time and down time. Since some of the states will not be visited at all, the value of $n_{states}$ is initialized to 70% of the total number states. The number of actions ‘$n_{action}$’ is initialized to $2^N - 1$. Then the permissible action set corresponding to each hour based on the load demand at that particular hour are identified. The algorithm during the learning phase proceeds as follows.

At each hour $k$, the state $x_k$ depending on the previous state and action is found as explained previously. The state $x_k$ is added to the set of states $\chi_k$ if not already present and find the index of the state $x_k$. From the permissible action set, one of the actions is chosen based on $\varepsilon$-greedy method. Then the next state $x_{k+1}$ can be found corresponding to stage $k+1$. On taking action $a_k$ the state of the system proceeds from $x_k$ to $x_{k+1}$ as given in (4.10). The reward, $g(x_k, a_k, x_{k+1})$ is given by equation (4.5).

The Q value corresponding to the particular hour ‘$k$’, action ‘$a_k$’ (decimal value of the binary string) and index no (ind_xk) is then updated using the equation:

$$Q^{n+1}(\text{ind}_x k, a_k) = Q^n(\text{ind}_x k, a_k) + \alpha \left[ g(x_k, a_k, x_{k+1}) + \gamma \min_{a' \in A_{k+1}} Q^n(\text{ind}_x_{k+1}, a') - Q^n(\text{ind}_x k, a_k) \right]$$

(4.11)

If the stage is the last one ($k = T$), corresponding to the last hour to be scheduled, there is no more succeeding stages and the updating equation reduces as,

$$Q^{n+1}(\text{ind}_x k, a_k) = Q^n(\text{ind}_x k, a_k) + \alpha \left[ g(x_k, a_k, x_{k+1}) - Q^n(\text{ind}_x k, a_k) \right]$$

(4.12)

At each episode of the learning phase, the algorithm passes through all the ‘$T$’ stages. As the algorithm completes several iterations, the estimated Q values will reach nearer to the optimum values and then the optimum schedule or allocation can be easily retrieved for each stage $k$ as, $a_k^* = \arg \min_{a_k \in A_k} (Q(\text{ind}_x k, a_k))$

The entire algorithm is illustrated as RL_UCP3.
Algorithm for Unit Commitment using state indexing (RL_UCP3)

Read Unit Data
Read the initial status of the units, \( x_0 \)
Read the Load forecast for next \( T \) hours
Initialize 'nstates' (number of states) and 'nactions' (number of actions)
Initialize \( Q^0 [\text{ind}_{-x_0} a_0] = 0 , \quad 0 < \text{ind}_{-x_k} <= \text{nstates}, 0 < a_k <= \text{nactions} \)
Find the set of permissible actions corresponding to each hour \( k \)
Initialize \( \varepsilon = 0.5 \) and \( a = 0.1 \)
For \( n=1 \) to max _ iteration
Begin
Read the initial status of the units \( x_0 \)
Add the state \( x_0 \) to set \( \chi_0 \)
For \( k=0 \) to \( T-1 \)
Do
Find the feasible set of actions \( \mathcal{A}_k \) corresponding to state \( x_k \) considering up and down times.
Choose an action using \( \varepsilon \)-greedy strategy from the feasible set of actions
Find the next state \( x_{k+1} \)
If \( x_{k+1} \) is present in \( \chi_{k+1} \) Get the index \( \text{ind}_{-x_{k+1}} \)
Else Add \( x_{k+1} \) in \( \chi_{k+1} \) and obtain index \( \text{ind}_{-x_{k+1}} \)
Calculate cost as \( g(x_0, a_0, x_{k+1}) \)
If \( ( k != T-1) \) Update \( Q \) value using equation (4.11)
Else Update \( Q \) value using equation (4.12)
\( \text{ind}_{-x_k} = \text{ind}_{-x_{k+1}}. \)
End do
Update the value of \( \varepsilon \)
End
Save \( Q \) values.
This algorithm can accommodate the minimum up time and down time constraints easily, when the number of generating units is small. Up to 5 hours of minimum up time and minimum down time the algorithm is found to work efficiently. But when the minimum up time and minimum down time increase beyond 5 hours and the number of generating units is beyond six, the number of states visited increases. Then the number of Q values stored and updated becomes enormously larger. This demands more computer memory. In order to solve this issue and make an efficient solution, in the next section a state aggregation method is discussed which needs much less computer memory than the above formulated algorithm.

4.10 Reinforcement Learning algorithm for UCP, through State Aggregation (RL_UCP4)

While looking into the Unit Commitment Problem with minimum up time and minimum down time constraints, the state space become very huge. The huge state space is difficult to handle in a straight forward manner when the minimum up time / minimum down time increases or the number of generating unit increases. Storing of Q value corresponding to each state – action pair becomes computationally expensive. Some method is to be thought of to reduce the number of Q values to be handled. In the perspective of Unit Commitment problem one can group the different states having the same characteristics so that the goodness of the different groups is stored instead of goodness of the different states corresponding to an action. The grouping of states can be done based on the number of hours a unit has been UP or DOWN.

(i) A machine which has been already UP for duration equal to or greater than the minimum up time can be considered as to occupy a state ‘can be shut down’.

(ii) A unit which is already UP but not have covered minimum up time can be considered as to represent a state ‘cannot be shut down’.
(iii) An already offline unit which has been DOWN for number of hours equal to or more than its minimum down time can be represented as a state 'can be committed'.

(iv) A unit which has been DOWN but has not covered the minimum down time so that cannot be committed in the next immediate slot of time can be represented as a state 'cannot be committed'.

Thus, at any slot of time, each of the generating unit will be in any of the above mentioned four representative states. If these four conditions are denoted as decimal integers (0, 1, 2, 3), regardless of the UP time and DOWN time of a generating unit, the state is represented by one of this integer value. By aggregating the numerous states visited in the previous algorithm to a limited number of states, number of Q values to be stored and updated in the look up table is greatly reduced.

With the decimal numbers 0,1,2,3 representing the aggregated states of a unit, for an N generating unit system the state $x_k$ is represented as a string of integers having length $N$ and with each integer having any of these four values. Then the state can be represented as a number with base value 4. For an N generating unit problem, there will be $4^N-1$ possible states, regardless of minimum up time and down time of the different units. (In the previous algorithm RL_UCP3, the number of states increases with increase in the minimum up time and / or down time). This reduction in the number of states drastically reduces the size of look up table for storing the Q values. Now an algorithm is formulated making use of state aggregation technique for handling the up/down constraints of the units.

The number of states, $nstates$ is initialized to $4^N-1$ and the number of actions $naction$ to $2^{N-1}$ for an N generating unit system. At any stage $k$ of MDP, the state of the system is represented as a string of integers as in the previous algorithm, integer value representing the number of hours the unit has been up or down.
In order to store the Q values, the state $x_k$ is mapped into set of aggregate states. Each aggregate state,

$$ag_{x_k} = \{(k, [ag_{p_0}^k, ag_{p_1}^k, ..., ag_{p_{N-1}}^k]), ag_{p_i} \in \{0,1,2,3\} \}$$

From any state $x_k$ an action is selected using one of the exploration strategies. On selecting an action $a_k$, the status of the units will change as, $x_{k+1} = f(x_k, a_k)$ given by equation (4.10). From the above explained categorization of states, $ag_{p_i}^k$ can be found corresponding to any $x_k = (k, [p_0^k, p_1^k, ..., p_{N-1}^k])$ as:

- $p_i^k$ positive and $p_i^k \geq U_i$, $ag_{p_i}^k = 0$;
- $p_i^k$ positive and $p_i^k < U_i$, $ag_{p_i}^k = 1$;
- $p_i^k$ negative and $p_i^k \leq D_i$, $ag_{p_i}^k = 2$;
- $p_i^k$ negative and $p_i^k > -D_i$, $ag_{p_i}^k = 3$.

The reward function for the state transition is found using the cost evaluations of the different generating units using equation (4.5). For each of the states $x_k$ and $x_{k+1}$, the corresponding aggregate state representation is found as $ag_{x_k}$ and $ag_{x_{k+1}}$. Each action in the action space is represented as the decimal equivalent of the binary string. At each state $k$, estimated $Q$ value corresponding to the state – action pair ($ag_{x_k}$, $a_k$) is updated using the equation,

$$Q^{n+1}(ag_{x_k}, a_k) = Q^n(ag_{x_k}, a_k) + \alpha [g(x_k, a_k, x_{k+1})$$
$$+ \gamma \min_{a' \in A_{k+1}} Q^n(ag_{x_{k+1}}, a') - Q^n(ag_{x_k}, a_k)]$$

(4.13)

During the last hour, omitting the term to account future pay-off $Q$ value is updated using the equation,
\[Q^{n+1}(ag_{X_k}, a_k) = Q^n(ag_{X_k}, a_k) + \alpha [g(x_k, a_k, x_{k+1}) - Q^n(ag_{X_k}, a_k)]\]

(4.14)

After a number of iterations, learning converges and the optimum schedule or allocation for each state \(x_k\) can be easily retrieved after finding the corresponding aggregate state as,

\[a^*_k = \text{argmin}_{a \in A_k} \{ Q(ag_{X_k}, a) \}, k = 0, \ldots, T - 1.\]

The entire algorithm using state aggregation method is given below:

**Algorithm for Unit Commitment through state aggregation (RL_UCP4)**

*Read Unit Data*
*Read the Load forecast for next T hours.*
*Initialize nstates (number of states) and nactions (number of actions)*
*Initialize \(Q^0[ag_{X_b}, a_i] = 0, \forall ag_{X_b}, \forall a_i\)*
*Find the set of permissible actions corresponding to each hour k*
*Initialize the learning parameters*
*For n=1 to max_episode*
*Begin*
*Read the initial status of the units \(x_0\)*
*For k=0 to T-1*
*Do*
*Find aggregate state \(ag_{X_k}\) corresponding to \(x_k\)*
*Find the feasible set of actions \(A_k\) corresponding to state \(x_k\) considering up and down times.*
*Choose an action using \(\epsilon\)-greedy strategy from the feasible set of actions*
*Contd...*
Reinforcement Learning Approaches for Solution of Unit Commitment Problem

Find the next state $x_{k+1}$

Find the corresponding aggregate state $ag_{x_{k+1}}$ of $x_{k+1}$

Calculate the reward $g(x_k, a_k, x_{k+1})$

If $k \neq T-1$ Update $Q$ value using equation (4.13)

Else Update $Q$ value using equation (4.14)

End do

Update the value of $\epsilon$.

End

Save $Q$ values.

The optimal schedule $[a_0^*, a_1^*, ..., a_{T-1}^*]$ is obtained using policy retrieval steps similar to the algorithm given in section 4.8.

4.11 Performance Evaluation

Solution to Unit Commitment Problem has now been proposed by various Reinforcement Learning approaches. Now, one can validate and test the efficacy of the proposed methods by choosing standard test cases. The high level programming code for the algorithms is written in C language in Linux environment. The execution times correspond to Pentium IV, 2.9 GHz, 512 MB RAM personal computer.

In order to compare the $\varepsilon$ greedy and pursuit solutions (RL_UCP1 and RL_UCP2) a four generating unit system and an eight generating unit system are considered. The generation limits, incremental and start up cost of the units are specified. Performance of the two algorithms is compared in terms of number of iterations required in the learning phase and the computation time required for getting the commitment schedule.

In order to validate and compare the last two algorithms (RL_UCP3 and RL_UCP4), four generating unit system with different minimum up time and down time are considered. The schedule obtained is validated and the performance comparison is made in terms of execution time of the entire algorithm, including the learning and policy retrieval. In order to prove the scalability of the algorithms, an
eight generating unit system with given minimum up time and down time limits is also taken for case study.

For comparing with the recently developed stochastic strategies a ten generating unit system with different minimum up time and down time limits are taken for case study. The schedule obtained and the computation time is compared with two hybrid methodologies: Simulated Annealing with Local Search (SA LS) and Lagrange Relaxation with Genetic Algorithm (LRGA).

In order to apply the proposed Reinforcement Learning algorithms, first suitable values of the learning parameters are to be selected. Value of $\varepsilon$ balances the rate of exploration and exploitation. For balancing exploration and exploitation, a value of 0.5 is taken for the learning parameter $\varepsilon$ initially. In every $(\text{max}_{\text{iteration}}/10)$ iterations, $\varepsilon$ is reduced by 0.04 so that in the final phases, $\varepsilon$ will be 0.1.

Discount parameter $\gamma$ accounts for the discount to be made in the present state in order to account of future reinforcements and since in this problem, the cost of future stages has the same implication as the cost of the current stage, value of $\gamma$ is taken as 1. The step size of learning is given by the parameter $\alpha$ and it affects the rate of modification of the estimate of $Q$ value at each iteration step. By trial and error $\alpha$ is taken as 0.1 in order to achieve sufficiently good convergence of the learning system. The RL parameters used in the problem are also tabulated in Table 4.1.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0.5</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>1</td>
</tr>
</tbody>
</table>

Now the different sample systems and load profile are considered, for evaluating the performance of the algorithms.
Case I – A four generating unit system

Consider a simple power system with four thermal units (Wood and Wollenberg [2002]). For testing the efficacy of the first two algorithms and to compare them, minimum up and down times are neglected. Load profile for duration of 8 hours is considered and is given in Table 4.2

Table 4.2 – Load profile for eight hours

<table>
<thead>
<tr>
<th>Hour</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load(MW)</td>
<td>450</td>
<td>530</td>
<td>600</td>
<td>540</td>
<td>400</td>
<td>280</td>
<td>290</td>
<td>500</td>
</tr>
</tbody>
</table>

The cost characteristics of the different units are taken to follow a linear incremental cost curve. That is, cost of generating a power $P_t$ by the $t^{th}$ unit is given as,

$$C_t(P_t) = N L_t + I C_t * P_t$$

where $NL_t$ represents the No Load cost of the $t^{th}$ unit and $IC_t$ is the Incremental cost of the same unit. The values $P_{min}$ and $P_{max}$ represent the minimum and maximum values of power generation possible for each of the units. The different unit characteristics and the generation limits are given in Table 4.3.

Table 4.3 – Generating Unit Characteristics

<table>
<thead>
<tr>
<th>Unit</th>
<th>$P_{min}$(MW)</th>
<th>$P_{max}$(MW)</th>
<th>Incremental cost Rs.</th>
<th>No Load Cost Rs.</th>
<th>Startup Cost Rs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>75</td>
<td>300</td>
<td>17.46</td>
<td>684.74</td>
<td>1100</td>
</tr>
<tr>
<td>2</td>
<td>60</td>
<td>250</td>
<td>18</td>
<td>585.62</td>
<td>400</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>80</td>
<td>20.88</td>
<td>213</td>
<td>350</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>60</td>
<td>23.8</td>
<td>252</td>
<td>0.02</td>
</tr>
</tbody>
</table>
When the learning is carried out using the first two algorithms, learning is first carried to find the maximum number of iterations required for the learning procedure.

In the learning procedure, 100 consecutive iterations are examined for modification in the estimated Q values. If the change is negligibly small in all these 100 iterations, the estimated Q values are regarded as optimum corresponding to a particular state – action pair. The iteration number thus obtained is approximated to nearest multiple of 100 is taken as ‘maximum iteration (max_iteration)’ and used in next trials. Different successive executions of the algorithm with max_iteration provided with almost the same results with tolerable variation.

RL_UCP1 indicated the convergence after 5000 iterations. While using RL_UCP2 the optimum is reached in 2000 iterations. The schedule obtained is given in Table 4.4 which is same as given in Wood and Wollenberg [2002].

Table 4.4 – Commitment schedule obtained

<table>
<thead>
<tr>
<th>Hour</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>0011</td>
<td>0011</td>
<td>1011</td>
<td>0011</td>
<td>0011</td>
<td>0001</td>
<td>0001</td>
<td>0011</td>
</tr>
</tbody>
</table>

The computation time taken by RL_UCP1 was 15.62 sec while that taken by RL_UCP2 was only 9.45sec. From this, it can be inferred that, the pursuit method is faster.

Case II – Eight generating unit system

Now an eight generating unit system is considered and a load profile of 24 hours is taken into account in order to prove the scalability of the algorithms. The load profile is given in Table 4.5
Table 4.5 – Load profile for 24 hours

<table>
<thead>
<tr>
<th>Hour</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load(MW)</td>
<td>450</td>
<td>530</td>
<td>600</td>
<td>540</td>
<td>400</td>
<td>280</td>
<td>290</td>
<td>500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hour</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>
</tr>
</thead>
<tbody>
<tr>
<td>Load(MW)</td>
<td>450</td>
<td>530</td>
<td>600</td>
<td>540</td>
<td>400</td>
<td>280</td>
<td>290</td>
<td>500</td>
</tr>
</tbody>
</table>

The cost characteristics are assumed to be linear as in previous case. The generation limit and the cost characteristics are given in Table 4.6.

Table 4.6 – Gen. Unit characteristics for Eight generator system

<table>
<thead>
<tr>
<th>Unit</th>
<th>P min (MW)</th>
<th>P max (MW)</th>
<th>Incremental Cost Rs.</th>
<th>No. Load Cost Rs.</th>
<th>Startup Cost Rs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>75</td>
<td>300</td>
<td>17.46</td>
<td>684.74</td>
<td>1100</td>
</tr>
<tr>
<td>2</td>
<td>75</td>
<td>300</td>
<td>17.46</td>
<td>684.74</td>
<td>1100</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>250</td>
<td>18</td>
<td>585.62</td>
<td>400</td>
</tr>
<tr>
<td>4</td>
<td>60</td>
<td>250</td>
<td>18</td>
<td>585.62</td>
<td>400</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>80</td>
<td>20.88</td>
<td>213</td>
<td>350</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>80</td>
<td>20.88</td>
<td>213</td>
<td>350</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>60</td>
<td>23.8</td>
<td>252</td>
<td>0.02</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>60</td>
<td>23.8</td>
<td>252</td>
<td>0.02</td>
</tr>
</tbody>
</table>

The optimal cost obtained for 24 hour period is Rs. 219,596/- and the solution obtained is given in Table 4.7. The status of the different units is expressed by the decimal equivalent of the binary string. For example during the first hour, the scheduled status is '3', which indicate 0000 0011.
Table 4.7 Commitment schedule for 24 hours

<table>
<thead>
<tr>
<th>Hour</th>
<th>0</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>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>3</td>
<td>3</td>
<td>131</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>134</td>
<td>6</td>
</tr>
<tr>
<td>Hour</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>Status</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>70</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Comparison of two algorithms RL_UCP1 and RL_UCP2 are given in Table 4.8. The number of iterations as well as computation time is again found to be less for the pursuit method when compared with e-greedy method.

Table 4.8- Comparison of algorithms RL_UCP1 and RL_UCP2

<table>
<thead>
<tr>
<th></th>
<th>RL_UCP1</th>
<th>RL_UCP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Unit system</td>
<td>5000</td>
<td>2000</td>
</tr>
<tr>
<td>No: of iterations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(sec.)</td>
<td>15.62</td>
<td>9.45</td>
</tr>
<tr>
<td>8 Unit system</td>
<td>$10^6$</td>
<td>$5 \times 10^5$</td>
</tr>
<tr>
<td>No: of iterations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(sec.)</td>
<td>34</td>
<td>17</td>
</tr>
</tbody>
</table>

Scalability of the proposed algorithms are now proved for simple Unit Commitment Problem.

Case III (Four unit system with minimum up time and minimum down time considered)

In order to validate the algorithms RL_UCP3 and RL_UCP4, first consider the four generator system (Wood and Wollenberg [2002]) with the given minimum up time and minimum down time. The unit characteristics and load profile are the same as given Table 4.2 and 4.3. The different units require different number of hours as minimum up time and minimum down time. The minimum up time, min. down time and the initial status of the units are given in Table 4.9.
Reinforcement Learning Approaches for Solution of Unit Commitment Problem

Table 4.9 – Minimum up time and minimum down time, initial status

<table>
<thead>
<tr>
<th>Unit</th>
<th>Min. Up time (Hr.)</th>
<th>Min. Down time (Hr.)</th>
<th>Initial Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4</td>
<td>-4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

The initial status -1 indicate that the particular unit has been DOWN for 1 hour and the initial status 1 represent that the unit has been UP for 1 hour.

The learning of the system is carried out using RL_UCP3 and RL_UCP4. A number of states are visited and after $10^5$ iterations, the Q values approach optimum. RL_UCP3 enumerates and stores all the visited states. The goodness of each state action pair is stored as Q value. On employing state aggregation in RL_UCP4, the number of entries in the stored look up table is reduced prominently. This is reflected by the lesser computation time. The optimum schedule obtained is tabulated in Table 4.10 which is consistent with that given through Dynamic Programming (Wood and Wollenberg [2002])

Table 4.10 – Optimum schedule obtained

<table>
<thead>
<tr>
<th>Hour</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0110</td>
</tr>
<tr>
<td>2</td>
<td>0110</td>
</tr>
<tr>
<td>3</td>
<td>0111</td>
</tr>
<tr>
<td>4</td>
<td>0110</td>
</tr>
<tr>
<td>5</td>
<td>0110</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0110</td>
</tr>
<tr>
<td>8</td>
<td>0110</td>
</tr>
</tbody>
</table>

In RL_UCP3, since the number of states and hence the number of Q values manipulated are more, the execution time is more. In case of RL_UCP4, the number of states is drastically reduced due to the aggregation of states and hence the schedule is obtained in much lesser time. Time of execution of the algorithms RL_UCP3 and RL_UCP4 are tabulated for comparison in Table 4.11
Table 4.11 – Comparison of RL_UCP3 and RL_UCP4

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Execution Time (Sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RL_UCP3</td>
<td>9.68</td>
</tr>
<tr>
<td>RL_UCP4</td>
<td>3.89</td>
</tr>
</tbody>
</table>

From the comparison of execution time, it can be seen that state aggregation has improved the performance very much.

**Case IV – Ten generating unit system**

In order to prove the flexibility of RL_UCP4 and to compare with other methods, next a ten generating unit with different initial status given is considered (Cheng *et al.* [2000]).

In this case minimum up time and minimum down time are also different for different units. Minimum up time of certain units is 8 hours, which is difficult to be handled by RL_UCP3. The cost functions are given in quadratic cost form, \( C(P) = a + bP + cP^2 \), where \( a \), \( b \) and \( c \) are cost coefficients and \( P \) the power generated. The values of the cost coefficients \( a \), \( b \) and \( c \) for the different generating units are given in Table 4.12. For a load profile of eight hours given in Table 4.13, the algorithm gave an optimum result in \( 2 \times 10^5 \) iterations. The obtained commitment schedule is given in Table 4.14.
### Table 4.12 – Generating Unit characteristics of 10 generator system

<table>
<thead>
<tr>
<th>Unit</th>
<th>P min (MW)</th>
<th>P max (MW)</th>
<th>Pmin (MW)</th>
<th>Pmax (MW)</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>Initial status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>150</td>
<td>455</td>
<td>1000</td>
<td>16.19</td>
<td>0.00048</td>
<td>4500</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>150</td>
<td>455</td>
<td>970</td>
<td>17.26</td>
<td>0.00031</td>
<td>4000</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>130</td>
<td>700</td>
<td>16.6</td>
<td>0.00211</td>
<td>550</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>130</td>
<td>700</td>
<td>16.6</td>
<td>0.002</td>
<td>360</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>160</td>
<td>450</td>
<td>19.7</td>
<td>0.00031</td>
<td>300</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>85</td>
<td>370</td>
<td>22.26</td>
<td>0.0072</td>
<td>340</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
<td>85</td>
<td>480</td>
<td>27.74</td>
<td>0.00079</td>
<td>520</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>55</td>
<td>660</td>
<td>25.92</td>
<td>0.00413</td>
<td>60</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>55</td>
<td>665</td>
<td>27.37</td>
<td>0.00222</td>
<td>60</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>55</td>
<td>670</td>
<td>27.79</td>
<td>0.00173</td>
<td>60</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 4.13 – Load profile for 24 hour

<table>
<thead>
<tr>
<th>Hour</th>
<th>P_Load (MW)</th>
<th>Hour</th>
<th>P_Load (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>700</td>
<td>13</td>
<td>1400</td>
</tr>
<tr>
<td>2</td>
<td>750</td>
<td>14</td>
<td>1300</td>
</tr>
<tr>
<td>3</td>
<td>850</td>
<td>15</td>
<td>1200</td>
</tr>
<tr>
<td>4</td>
<td>950</td>
<td>16</td>
<td>1050</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>17</td>
<td>1000</td>
</tr>
<tr>
<td>6</td>
<td>1100</td>
<td>18</td>
<td>1100</td>
</tr>
<tr>
<td>7</td>
<td>1150</td>
<td>19</td>
<td>1200</td>
</tr>
<tr>
<td>8</td>
<td>1200</td>
<td>20</td>
<td>1400</td>
</tr>
<tr>
<td>9</td>
<td>1300</td>
<td>21</td>
<td>1300</td>
</tr>
<tr>
<td>10</td>
<td>1400</td>
<td>22</td>
<td>1100</td>
</tr>
<tr>
<td>11</td>
<td>1450</td>
<td>23</td>
<td>900</td>
</tr>
<tr>
<td>12</td>
<td>1500</td>
<td>24</td>
<td>800</td>
</tr>
</tbody>
</table>
**Table 4.14 – Unit Commitment schedule**

<table>
<thead>
<tr>
<th>Hr</th>
<th>Load (MW)</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>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>700</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>
</tr>
<tr>
<td>2</td>
<td>750</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>
</tr>
<tr>
<td>3</td>
<td>850</td>
<td>1</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>
</tr>
<tr>
<td>4</td>
<td>950</td>
<td>1</td>
<td>1</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>1000</td>
<td>1</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>1100</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>1150</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>8</td>
<td>1200</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>9</td>
<td>1300</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>1400</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>
</tr>
<tr>
<td>11</td>
<td>1450</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>
</tr>
<tr>
<td>12</td>
<td>1500</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>
</tr>
<tr>
<td>13</td>
<td>1400</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>
</tr>
<tr>
<td>14</td>
<td>1300</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>
</tr>
<tr>
<td>15</td>
<td>1200</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>16</td>
<td>1050</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>17</td>
<td>1000</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>18</td>
<td>1100</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>19</td>
<td>1200</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>20</td>
<td>1400</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>
</tr>
<tr>
<td>21</td>
<td>1300</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>
</tr>
<tr>
<td>22</td>
<td>1100</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>900</td>
<td>1</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>
</tr>
<tr>
<td>24</td>
<td>800</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>
</tr>
</tbody>
</table>
By executing RL_UCP4 for the above characteristics, the commitment schedule is obtained in 268 sec. The obtained schedule is given in Table 4.14. The cost obtained and the computation time are compared with that obtained through hybrid methods using Lagrange Relaxation and Genetic Algorithm (LRGA) proposed by Cheng et al. [2000] and Simulated Annealing and Local search (SA LS) suggested by Purushothama and Lawrence Jenkins [2003]. Comparison of the cost and time are given 4.15.

**Table 4.15 – Comparison of cost and time**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cost(Rs.)</th>
<th>Execution Time(sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRGA</td>
<td>564800</td>
<td>518</td>
</tr>
<tr>
<td>SA LS</td>
<td>535258</td>
<td>393</td>
</tr>
<tr>
<td>RL_UCP4</td>
<td>545280</td>
<td>268</td>
</tr>
</tbody>
</table>

A graphical representation of Table 4.15 is given in 4.1

![Fig 4.1 Comparison of execution time (sec.) of RL approach with other methods](image)

---

113
The comparison revealed that the proposed method gave comparable cost with other methods and takes less computation time. Quick decision making is advantageous in a practical power system scheduling since the economy of power generation directly rely on the same. When a practical system is considered the cost is not at all constant and it changes from time to time. Reinforcement Learning provides with a solution for handling the same.

4.12 Conclusion

Unit commitment has now been formulated as a multi stage decision making task and then Reinforcement Learning based solution strategies are developed for solving the same. First the minimum up time and minimum down time limitations are neglected and the scheduling policy is arrived using $\varepsilon$ greedy and pursuit methods for action selection. Then the minimum up time and minimum down time of the generating units are considered to find the optimum commitment schedule. An index based approach has been formulated to reduce the size of the search space in RL_UCP3 and state aggregation method is implemented to make the solution more efficient and reduce computation time in RL_UCP4. All the four algorithms have been verified for different systems with given load profile. The results seem to be good with respect to the computation time when compared with some of the recent methods.

Getting a profitable schedule based on the instantaneous cost is a much needed task. This indeed necessitates algorithms which provide the optimum schedule at minimum time as far as possible. In a practical power system, at the load dispatch centre time to time decision making is a serious task since the impact on the economy is very serious. A quick decision making is necessitated at each change of the stochastic data. As the Reinforcement Learning method proves to be faster and more efficient in handling system data compared to other strategies, it is much suitable for practical systems.