Electrical Power and Energy Systems 33 (2011) 836–845

Contents lists available at ScienceDirect

Electrical Power and Energy Systems

journal homepage: www.elsevier.com/locate/ijepes

Reinforcement Learning approaches to Economic Dispatch problem

E.A. Jasmin [a,*], T.P. Imthias Ahamed [b], V.P. Jagathy Raj [c]

[a] *Dept. of Electrical & Electronics, Govt. Engineering College, Thrissur, Kerala, India*
[b] *T.K.M. College of Engineering, Kollam, Kerala, India*
[c] *Cochin University of Science and Technology, Kochi, Kerala, India*

ARTICLE INFO

Article history:
Received 29 June 2007
Received in revised form 6 July 2009
Accepted 9 December 2010
Available online 5 February 2011

Keywords:
Economic Dispatch
Power generation control
Reinforcement Learning
Q learning

ABSTRACT

This paper presents Reinforcement Learning (RL) approaches to Economic Dispatch problem. In this paper, formulation of Economic Dispatch as a multi stage decision making problem is carried out, then two variants of RL algorithms are presented. A third algorithm which takes into consideration the transmission losses is also explained. Efficiency and flexibility of the proposed algorithms are demonstrated through different representative systems: a three generator system with given generation cost table, IEEE 30 bus system with quadratic cost functions, 10 generator system having piecewise quadratic cost functions and a 20 generator system considering transmission losses. A comparison of the computation times of different algorithms is also carried out.

1. Introduction

Reinforcement Learning Problem in general is a problem of learning from interaction to achieve a specified goal. The learner (decision maker) continuously interacts with the environment (system). The interaction is through actions and associated rewards. The agent performs an action from the permissible set of actions at the particular state of the environment. The environment gives back a numerical reward which is a measure of the goodness of the performed action. Such a learning scheme is widely employed in solving several difficult problems such as control of inverted pendulum, Playing Backgammon and other computer games, Elevator control [1–5], etc. There are few applications of Reinforcement Learning in Power System problems also. It has been applied for Load Frequency Control of generators [6], Unit Commitment problem [7], Power system transient stability enhancement [8], Auction Based Pricing [9], Optimal bidding of a Genco [10].

As far as power generation control is considered, it is basically having three time based control loops: Unit Commitment, Economic Dispatch and Load Frequency Control [13]. In Economic Dispatch problem, cost of generation of power has been represented in a variety forms including cost tables, quadratic functions, etc. For getting more accurate representation, cost functions are also sometimes expressed as piecewise quadratic functions. For solving this scheduling problem, so many computational and intelligent techniques have been developed so far. Some of the strategies applied for solution of this problem are explained in [14]. Fast computation Hopfield neural network along with dynamic programming is used for getting the schedule of generation in [15]. The work presented in [16] explains the application of Hopfield Neural Network for Economic and Emission Load Dispatch. The work presented in [17] also gives out the idea of using Hopfield Neural Network as a tool for solving this control problem. [18] and [19] considers the security constraints of the problem also and is solved through decomposition and coordination algorithms and variable scaling hybrid differential programming method respectively. [20] uses an immune based method known as Clonal algorithm. Using Radial Basis Function Network to compute optimum value for lambda and then using lambda iteration method the problem is solved in [21]. Simple Genetic algorithm is used for finding optimum dispatch [22] and simulated annealing is used as tool in [23]. Piece wise quadratic functions are considered and solution is made through an improved genetic algorithm in [24]. Recently a direct search method viewing the non convex nature of economic dispatch problem is employed in the work presented in [25] and [26] also gives out a direct search approach. Different kinds of efficient evolutionary algorithms are developed in [27] and [28]. Particle swarm optimization is the technique used by the researchers in [29] and [30] while a new optimization based algorithm: 'Taguchi method' is used in [31].

Reinforcement Learning seems to provide better flexibility and easiness in accommodating randomness in the cost strategies associated with complex systems. Also since this learning strategy relies on an evaluative feedback approach, like other intelligent techniques, it can work on systems with ill defined models. In

* Corresponding author. Tel.: +91 9495465409.
E-mail address: eajasmin@gmail.com (E.A. Jasmin).

0142-0615/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ijepes.2010.12.008

addition, the evaluation of reward corresponding to an action which is the core part of this strategy need not be based on mathematical functions. It just needs a numerical value indicating the goodness of a particular schedule. In this approach, we treat Economic Dispatch as a multi stage decision making problem which distributes the power demand among the generating units available on line in such a way as to minimize operating cost. Recently it was shown that RL algorithms could be applied to solve Economic Dispatch problem by considering a simple three generator system having quadratic cost functions [11,12]. In this paper a class of RL algorithms are proposed. Performance analysis of the algorithms are carried out considering different test cases with different types of cost functions.

The organization of the rest of the paper is as follows. In Section 2, Economic Dispatch problem is presented in terms of mathematical equations. Section 3 gives an explanatory discussion on Reinforcement Learning and introduces the terms and notations used in RL algorithm using the example of shortest path problem. In Section 4 Economic Dispatch is formulated as a multi stage decision making problem. Section 5 explains the Reinforcement Learning algorithms for solution of this decision making problem. Transmission losses are also considered and algorithm for solving the same is presented in Section 6. Performance evaluation of the proposed solution methods using the different case studies are given in Section 7. Conclusion with discussion on future improvements possible follows.

2. Economic Dispatch

Economic Dispatch is basically to find the relative loadings of the various generating units online.

The allotment should be in such a way that the cost of generation should be minimum as far as possible. At the same time generating unit power constraints should also be met. Therefore the objective function of Economic Dispatch problem F_T is equal to the total cost for supplying the demanded load P_T. The problem is to minimize F_T subject to the constraints that the total generated power and the demanded load equals and the power constraints on all units are being met. Mathematically the objective function F_T can be expressed as

$$F_T = F_1(P_1) + F_2(P_2) + F_3(P_3) + \dots F_N(P_N) = \sum_{i=1}^{N} F_i(P_i) \tag{1}$$

where $F_i$denotes the cost function of ith unit and P_i the electrical power generated by that particular unit and constraints are

$$P_T - \sum_{i=1}^{N} P_i = 0 \tag{2}$$

$$P_{\min i} < P_i < P_{\max i}, \quad \text{for } i = 1 \text{ to } N \tag{3}$$

where P_T, total load power demand; $P_{\min\ i}$, min. power generation of ith unit; $P_{\max\ i}$, max. power generation of ith unit.

Therefore the problem is to minimize the cost function F_T subject to the constraints given in Eqs. (2) and (3).

3. Reinforcement Learning

Reinforcement Learning is one effective method in the solution of multi stage decision making problems. For a comprehensive study of the subject, refer [1,2,32,33]. To make the paper self contained, a brief introduction is given here.

A general layout of Reinforcement Learning task is given in Fig. 1.

The agent interacts with the environment through actions and associated rewards. It uses training information that evaluates actions (in terms of reward received from the environment on performing an action) taken by the agent. The effect of action at any instant depends on nature of the problem environment.

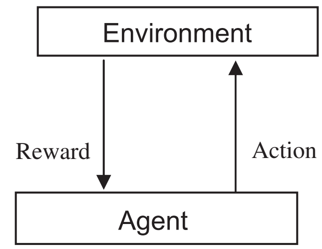

Fig. 1. A general layout of Reinforcement Learning task.

In order to make the concepts clear and to explain the steps followed in the learning task let us consider the example of a simple shortest path problem. Consider a grid world problem shown in Fig. 2.

The grid considered is having 36 cells arranged in 6 rows and 6 columns. A robot can be at any one of the possible cells at any instant. We can refer the cell number as state of the robot. 'G' denotes the goal state to which the robot aim to reach and the crossed cells denote cells with some sort of obstacles. There is cost associated with each cell transition while the cost of passing through a cell with obstacle is much higher compared to other cells. Starting from any initial position in the grid, robot can reach the goal cell by following different paths and correspondingly cost incurred will also vary. The problem is to find an optimum path to reach the goal state (cell) starting from any one of the initial state. We can denote the state of the robot at instant k as x_k. At any instant k, the robot can take any of the action (movement) a_k, from the set of permissible actions in the action set A_{x_k} which also depend on the current state x_k. For example if $x_k = 7$, $A_{x_k} = \{\text{right}, \text{up}, \text{down}\}$ and if $x_k = 1$, $A_{x_k} = \{\text{right}, \text{down}\}$. The state occupied by the robot in $k + 1$, x_{k+1} depends on x_k and a_k. That is,

$$x_{k+1} = f(x_k, a_k) \tag{4}$$

For example, if $x_k = 7$ and a_k = Down then $x_{k+1} = 13$ while when a_k = Up, $x_{k+1} = 1$. The state x_{k+1} can be found from the simulation model of the grid or studying the environment in which robot moves.

Therefore the shortest path problem can be stated as finding the sequence of actions $a_0, a_1, \dots, a_{T-1}$ starting from any initial state such that the total cost for reaching goal state G is minimum. The numerical cost when the agent (robot) performs an action a_k at state x_k making transition to x_{k+1} is denoted as r_k, known as immediate reward or reinforcement. This reward r_k in general can depend on current state, action and the next state. That is,

$$r_k = g(x_k, a_k, x_{k+1}) \tag{5}$$

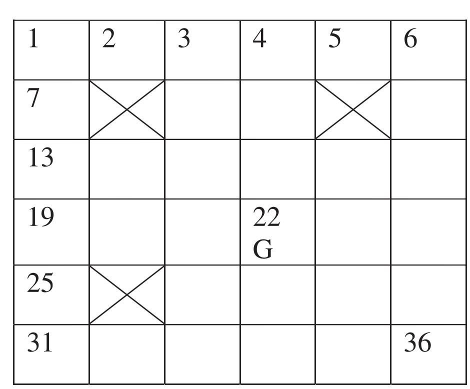

Fig. 2. Grid world problem.

In this simple grid world we assume a cost of 1 unit for transition to an ordinary cell while 10 units for a cell having some obstacles.

$$g(x_k, a_k, x_{k+1}) = 1, \quad \text{if } x_{k+1} \text{ is a normal cell}$$
$$= 10, \quad \text{if } x_{k+1} \text{ is a cell with obstacles}$$

To find the total cost, the cost on each transition can be cumulated. Now the total cost for reaching the goal state can be taken as $\sum_{k=0}^{T-1} g(x_k, a_k, x_{k+1})$, x_0 being the initial state and T being the no. of transitions to reach the goal state. The action selection can be considered as following a policy π. i.e., $\pi(x)$ denotes the action taken on reaching the state x. The Reinforcement Learning task now reduces to finding an optimum policy π^* which gives out the optimum action at each state (cell) in order to get minimum total cost. For this problem, the aim is to minimize the total cost which is equal to $\sum_{k=0}^{T-1} g(x_k, a_k, x_{k+1})$.

In general, this need not be the case. Here we are assuming cost incurred at all stages have equal importance. However in some cases a cost of 100$ incurred at a later stage may not have the same effect as a cost of 100$ at the current stage. Hence future cost may be discounted by a factor. In such situations, total cost is defined as,

$$\text{cost} = \sum_{k=0}^{T-1} \gamma^k g(x_k, a_k, x_{k+1})$$

In a more general setting, cost incurred may be a random variable. In such a problem the objective function is $E\left\{\sum_{k=0}^{T-1} \gamma^k g(x_k, a_k, x_{k+1})\right\}$ which is the expected cost.

Now we introduce Q value. Q value of a multi stage problem is defined as:

$$Q^\pi(x, a) = E^\pi \sum_{k=0}^{\infty} \gamma^k r_k,$$

$Q^\pi(x,a)$ is the expected long term reinforcement when we start in state (cell) x, take action a in the starting state and thereafter follow the policy π and γ is the discount factor. For example $Q(2, D)$ denote the Q value associated with the cell 2 and starting with action *Down*. Similarly each of the different cells and possible actions will be associated with a particular Q value. Now define $Q^\pi(x_k, a_k)$ as the expected value of the total cost incurred in taking an action a_k from the state x_k and thereafter taking actions based on policy $\boldsymbol{\pi}$. One can call $Q^*(x_k, a_k)$ as an optimal Q function if $Q^*(x_k, a_k) = \min_\pi Q^\pi(x_k, a_k)$. Therefore, now the optimal policy can be stated as: $\pi^*(x_k) = \arg\min_{a_k \in Ax_k}\{Q^*(x_k, a_k)\}$, $a_k \in Ax_k$ being the action to be taken at state x_k in order to get minimum cost.

Therefore if one can find $Q^*(x_k, a_k)$ for all state–action pairs, then the path for reaching the goal state can be traced out from any initial state at minimum cost. In Q learning algorithm, first all Q values are initialized with some initial value, $Q^0(x_k, a_k)$. At each iteration n, on reaching x_k, an action a_k is taken based on the current estimate of $Q^*(x_k, a_k)$ i.e., $Q^n(x_k, a_k)$. Once action is taken at state x_k, it makes transition to x_{k+1} and the reward $g(x_k, a_k, x_{k+1})$ can be found from simulation. The Q value is updated using the equation,

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

$0 < \alpha < 1$ is a constant and is called step size of learning.

One issue to be settled is how to choose action in each step. At the nth iteration, Q^* is unknown but Q^n (estimate of Q^*) is known to the learning agent. If Q^n is same as Q^*, the best action is, $a_g = \arg\min_{a \in A_k}\{Q^n(x_k, a)\}$. This way of choosing action is called greedy algorithm and a_g is called greedy action. If one chooses greedy action initially, it may be wrong if Q^n is not equal to Q^*.

One choice is to go for ε – greedy algorithm. Here Greedy action a_g will be chosen with a probability of $(1 - \varepsilon)$ and one among all other permissible actions is chosen with a probability of ε. ε can be taken as fixed value throughout the learning. But a small fixed value may slow down the learning process, while a large fixed value may make the learning process not to converge. To overcome these problems, an adaptive method is used for fixing of exploration rate. In this approach ε is fixed arbitrarily at some sufficiently large value and then progressively reduced to make a smooth and fast convergence possible.

The ε – greedy method discussed above provides a good method of action selection, for providing better exploration in the initial phases of learning while exploiting the goodness of greedy action during the later phases. However ε – greedy requires a gradual reduction of ε. That is, a proper cooling schedule is to be designed which gradually updates the value of ε as the learning proceeds so that proper convergence and correctness of the result are assured. The length of learning phase mainly depends on this cooling schedule and therefore it is one significant part of ε – greedy method. It is a difficult task to develop a good cooling schedule so as to ensure minimum time for convergence.

Another stochastic policy followed for selection of action in the Reinforcement Learning task is Pursuit algorithm. In this method along with maintaining estimates of Q values as measure of goodness of actions, some preference is also associated with actions. Each action a_k at any state x_k is having a probability $p_{x_k}(a_k)$ of being chosen. These probability values will be same for all actions and all states initially assuring sufficient exploration of the action space. Then on performing an action a_k at any state x_k during learning, the numerical reward is used to update the estimate of Q value associated with the state–action pair. Along with that, based on the current estimates of Q values, probability values associated with actions are also modified as.

$$p_{x_k}^{n+1}(a_k) = p_{x_k}^n(a_k) + \beta[1 - p_{x_k}^n(a_k)], \quad \text{when } a_k = a_g$$
$$p_{x_k}^{n+1}(a_k) = p_{x_k}^n(a_k) - \beta p_{xk}^n(a_k), \quad \text{when } a_k \neq a_g$$

where $0 < \beta < 1$ is a constant. Thus at each iteration n of the learning phase, algorithm will slightly increase the probability of choosing the greedy action a_g in state x_k and proportionally decrease the probability associated with all other possible actions. Initially since all probabilities are made equal, sufficient exploration of action space is assured. When the algorithm proceeds a number of iterations, with high probability $Q^n(x,a)$ will approach to $Q^*(x,a)$ corresponding to all states. This is because, when the parameter β is properly chosen, after sufficient number of iterations, the greedy action in state x, with respect to Q^n would be the same as greedy action in state x with respect Q^* which gives the optimal action. In other words, through the iterative updating of probabilities, probability of optimal action increases successively. This in turn indicates an increase in probability of selecting the optimum action in the succeeding steps. If α and β are sufficiently small, $p_x^n(\pi^*(x))$ would converge with high probability to unity.

In both these methods of solution, the problem is made to start from any of the initial states at random at each of the iteration of learning so that it goes through different transition paths updating the corresponding Q values [1]. In the next sections, solution to Economic Dispatch is obtained using the above two methods for exploration of action space.

4. Economic Dispatch as multi stage decision problem

To view Economic Dispatch as a multi stage decision making problem, the various stages of the problem are to be identified. Consider a system with N generating units committed for dispatch. Then Economic Dispatch problem involves deciding the amount of power to be dispatched by $G_0, G_1, G2, \ldots\ldots\ldots\ldots\ldots G_{N-1}$.

In this formulation the amount of power to be dispatched by G_k is denoted as action a_k. Action a_k in Reinforcement Learning terminology, corresponds to a power allocation $P_k MW$ to generating unit G_k. P_k is numerically same as a_k. Therefore, the action set A_k consists of the different values of power dispatch possible to G_k. That is, $A_k = \{Min_k, \ldots, \ldots, \ldots, Max_k\}$, Min_k being the minimum value of power that can be allotted to G_k and Max_k being the maximum power that can be allotted to G_k.Values of Min_k and Max_k depend on the minimum and maximum values of power generation possible with the kth unit and also maximum and minimum power that can be allotted among the remaining $N–k$ units available. Therefore, action set A_k is a dynamically varying one depending on the power already allotted to the previously considered units.

The quantization step (in MW power) is chosen based on the accuracy needed. A very small value is not necessitated due to the setting of the reference point setting in a plant. An optimum value is chosen based on the accuracy needed and the setting of the units. Also as number of generating units and hence the range of possible demand increases, the number of states in the state space increases. State space is also discretized to have definite number of states. For defining the same, an optimum value of step size is to be chosen so as to get the required accuracy keeping the number of states manageable.

Now the problem can be stated as follows: Initially there are N units and P_D MW of power to be dispatched. The initial stage is denoted as $stage_0$. In $stage_0$, a decision is made on how much power is to be dispatched by G_0. This action is denoted as a_0 and corresponds to P_0MW allocation to G_0.

On making this decision, stage1 is reached. Also $(P_D - a_0)$ MW of power remains to be allocated to the remaining *N-1* units $(G_1, G_2, \ldots, G_{N-1})$. In $stage_1$, a decision a_1 is made on how much power is to be dispatched to G_1. Similarly at stage k, a decision is made on how much power is to be dispatched by G_k. Thus, the stage $N-1$ is reached where the amount of power to be dispatched by $G_0, G_1, \ldots\ldots\ldots G_{N-2}$ are already decided as $(a_0, a_1, \ldots\ldots a_{N-2})$ which give directly the power allocations $(P_1, P_2, \ldots\ldots\ldots\ldots, P_{N-2})$.

From the power balance constraint (with $P_L = 0$), it follows that, $P_0 + P_1 + P_2 + \cdots\cdots\cdots P_{N-1} = P_D$ which directly implies that

$$P_{N-1} = P_D - (P_0 + P_1 + P_2 + \cdots\cdots\cdots\cdots P_{N-2})$$

Therefore, in $stage_{N-1}$, there is no choice but to allocate power P_{N-1}.

Each state at any $stage_k$ (k varies from 0 to $N-1$) can be defined as a tuple (k, D_k) where k is the number indicating the stage number and D_k, the power to be distributed among the remaining $N–k$ units.

That is, with $k = 0$, the state information is denoted as $(0, D_0)$ where D_0 is the load demand P_D for Economic Dispatch among the N generating units. The RL algorithm selects one among the permissible set of actions (between max. and min. power limits corresponding to one of the unit) and allocates to the particular machine considered so that it reaches the next stage ($k = 1$) with the remaining power after allocation, and $N-1$ units for generation. Transition from $(0, D_0)$ on performing an action $a_0 \in A_0$ results in the next state reached as $(1, D_1)$.

$$D_1 = D_0 - a_0$$

Or in general, in stage k, from state x_k on performing an action a_k reaches state x_{k+1}. i.e., state transition is from (k, D_k) to $(k+1, D_{k+1})$, where

$$D_{k+1} = D_k - a_k \tag{6}$$

This proceeds until the last stage. Therefore, state transition can be denoted as,

$$x_{k+1} = f(x_k, a_k)$$

'f' being the function of state transition defined by Eq. (6).

Thus, Economic Dispatch algorithm can be treated as one of finding an optimum mapping from the state space χ to action space A. The associative nature of the problem arises from the fact that each action denotes distribution of power to one unit so that the power to be distributed among the remaining units reduces by that much amount. Design of Economic Dispatch algorithm is finding or learning a good or optimal policy (allocation schedule) which is the optimum allocation at each stage. Such allocation can be treated as elements of an optimum policy π^*.

5. RL algorithms for Economic Dispatch

In the previous section, Economic Dispatch is formulated as a multi stage decision making problem. To find the best action corresponding to each state, Reinforcement Learning based solutions are explored. Solutions consist of two phases: learning phase and policy retrieval phase.

One major issue in the learning phase is how to develop an action selection strategy which can balance exploitation of available information and exploration of action space to learn new possibilities. There are various algorithms in RL literature [1,2,32,33] which balances exploration and exploitation. Here we develop RL algorithms for economic dispatch using two action selection strategies.

In algorithm I, a conceptually simple ε – greedy algorithm is used for action selection. In algorithm II Pursuit algorithm is used. In the next two section these learning algorithms are presented.

5.1. Algorithm I

For solving this multi stage problem using Reinforcement Learning, first step is fixing of state space χ, action space A and the immediate reinforcement function $g(x_k, a_k, x_{k+1})$ precisely. The different units can be considered arbitrarily as corresponding to the different stages.

Fixing of state space χ primarily depends on number of generating units available on line and the possible values of power demand (which in turn directly depends on min. and max. values of power generation possible with each unit). Since there are N stages for solution of the problem, the state space is also divided into N subspaces. Thus, if there are N units to be dispatched,

$$\chi = \chi_0 U \chi_1 U \ldots\ldots\ldots \chi_{N-1}.$$

The dispatch problem should go through N (no: of generating units) stages for making allocation to each of the N generating units. At any stage ($stage_k$), the part of state space to be considered (χ_k) consists of the different tuples having the stage number as k and power values varying from $D_{min(k)}$ to $D_{max(k)}$, $D_{min(k)}$ being the minimum power possible to be met with $N–k$ units and $D_{max(k)}$ the maximum power possible with $N–k$ units.

That is, $\chi_k = \{(k, D_{min(k)}), \ldots\ldots\ldots (k, D_{max(k)})\}$, where $D_{min(k)}$ = minimum power possible with $N–k$ units

$$= \sum_{i=k}^{i=N-1} P_{min(i)}$$

$$D_{max(k)} = \text{maximum power possible with } N-k \text{ units}$$

$$= \sum_{i=k}^{i=N-1} P_{max(i)}$$

At each step, the Economic Dispatch algorithm will select an action from the permissible set of discretised values and forward the system to one among the next permissible states.

The action set A_k consists of different values of MW power that can be allotted to k^{th} unit. The action set A_k depends on the demand value D_k at the current state x_k and also on the minimum and maximum power generation possible with remaining N–k units. Therefore action set A_k is dynamic in nature in the sense that it depends on the power already allotted up to that stage and also the minimum and maximum generation possible with the remaining N–k units. If D_k is the power to be allotted, minimum value and maximum value of action a_k are defined as

$$\begin{aligned} Min_k &= max[(D_k - D_{max(k)}), P_{min(k)}] \\ Max_k &= min[(D_k - D_{min(k)}), P_{max(k)}] \end{aligned} \tag{7}$$

The number of elements in these sets χ and A depends on the minimum and maximum limits and also the sampling step.

In Economic Dispatch problem, the reward function (g) can be chosen as the cost function itself. That is, reward received or cost incurred on taking an action a_k or allocating a power P_k at kth stage is the cost of generation of the power P_k. In the Reinforcement Learning terminology, the immediate reward,

$$r_k = g(x_k, a_k, x_{k+1}) = C_k(P_k) \tag{8}$$

Initially Q values of all state–action pairs are set to zero. In each iteration greedy action ($a_g = \arg\min_{a\in Ak}\{Qn(xk, a)\}$) is found. Then greedy action a_g will be chosen with a probability of $(1 - \varepsilon)$ and one among all other permissible actions is chosen with a probability of ε. As explained earlier based on the action selected system moves to the next state. Since the aim is to minimize the cost of generation estimated Q values of state–action pair are modified at each step of learning as,

$$Q^{n+1}(x_k, a_k) = Q^n(x_k, a_k) + \alpha[g(x_k, a_k, x_{k+1}) + \gamma \min_{a\in\mathscr{A}_{k+1}} Q^n(x_{k+1}, a) - Q^n(x_k, a_k)] \tag{9}$$

Here, α is the step size of the learning algorithm and γ is the discount factor.

When the system comes to the last stage of decision making, there is no need of accounting the future effects and then the estimate of Q value is updated using the equation,

$$Q^{n+1}(x_k, a_k) = Q^n(x_k, a_k) + \alpha[\gamma g(x_k, a_k, x_{k+1}) - Q^n(x_k, a_k) \tag{10}$$

As the learning steps are carried out sufficient number of times, estimated Q values of state–action pairs will approach to optimum so that we can easily retrieve the optimum policy (allocation) $\pi^*(x)$ corresponding to any state x.

The learning procedure can now be summarized. At each iteration of learning phase the algorithm will take the system initial condition (i.e., for k= 0) which is the power demand, as one random value within permissible limits. Then an action is performed which will allocate power to one of the units and then pass to the next stage (k = 1) with the remaining power. This proceeds until all the $N - 1$ units are allotted power. At each state transition step, the estimated Q value of the state–action pair is updated using Eq. (9).

As the learning reaches the last stage, since there is no choice of action, the remaining power to be allotted will be the power corresponding to the action ($a_{N-1} = D_{N-1}$). Then the Q value is updated using Eq. (10). The transition process is repeated a number of times (iterations) with random values of initial demand and each time the dispatch process goes through all the $N - 1$ stages. Value of ε is taken closer to 0.5 in the initial phases of learning and is reduced in every max_iteration/10 iterations by 0.04.

The entire algorithm of learning using ε greedy is given in Table 1:

Table 1
Algorithm for Economic Dispatch using ε greedy.

```
Get Unit parameters
Initialize the learning parameters
For all the stages Identify possible state vectors, χ_k
Evaluate minimum and maximum demands permissible at each stage
Initialize Q^0(x, a) to zero
Initialize ε = 0.5
For (n = 0 to max_iteration)
    Begin
        P_D=rand (Dmin_0, Dmax_0), D_0 = P_D
        For k = 0 to N − 2
        Do
            State tuple x_k = (k, D_k)
            Identify the action space A_k using Eq. (7)
            Select an action a_k from action set A_k
                using ε – greedy method.
            Apply action a_k and obtain the next state, D_k+1 = D_k − a_k
            Calculate the reward function using Eq. (8)
            Update Q^n to Q^n+1 using Eq. (9)
        End do
        a_N−1 = D_N−1
        Calculate the reward using Eq. (8)
        Update Q^n to Q^n+1 using Eq. (10)
        Update learning parameter ε.
    End.
    Save Q values.
```

5.2. Algorithm II

In this algorithm, we use pursuit for action selection in each step. In case of Pursuit algorithm, for any given state x_k, an action is selected based on the probability distribution function $p_{x_k}()$. In the case of Economic Dispatch, 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

$$p_{x_k}(a_k) = \frac{1}{n_k}$$

where n_k is total number of permissible actions at stage k. As in the previous algorithm, initialize the Q values of all state–action pairs to zero.

Then at each iteration step, an action a_k is selected based on the probability distribution. On performing action a_k, it reaches the next stage with $D_{k+1} = D_k - a_k$. The cost incurred in each step of learning is calculated as the sum of cost of producing power P_k with the kth generating unit. Q values are then updated using Eq. (9). At each of the iteration of learning, we find the greedy action as $a_g = \arg\min_{a\in\mathscr{A}_k}(Q(x_{k,}, a))$. Then accordingly the probabilities of actions in the action set are also updated as,

$$\begin{aligned} p^{n+1}_{x_k}(a_k) &= p^n_{x_k}(a_k) + \beta[1 - p^n_{x_k}(a_k)], \quad \text{when } a_k = a_g \\ p^{n+1}_{x_k}(a_k) &= p^n_{x_k}(a_k) - \beta[p^n_{x_k}(a_k)], \quad \text{when } a_k \neq a_g \end{aligned} \tag{11}$$

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. The entire algorithm is given in Table 2.

5.3. Policy retrieval algorithm

As the learning proceeds and updating of Q values of state–action pairs are done sufficiently large number of times, Q^n will be almost equal to Q^*. Then the learned Q values are used to obtain the optimum dispatch. For any value of power demand P_D, initialize $D_0 = P_D$. Then the state of the system is $(0, D_0)$. Find the greedy action at this stage as a_g which is the best allocation for 0th generating unit (P_0). The learning system reaches the next state as $(1, D_1)$ where $D_1 = D_0 - a_0$, find the greedy action corresponding to stage$_1$ as a_1. This proceeds up to $(N - 1)$th stage. Then a set of actions (allocations) $a_0,\ a_1,\ a_2, \ldots\ldots\ldots a_{N-1}$ is obtained which is the optimum schedule $P_0, P_1, \ldots\ldots\ldots P_{N-1}$ of generation corresponding to power P_D. The algorithm for getting the schedule from the learnt system follows:

Table 2
Algorithm for Economic Dispatch using Pursuit method.

Get Unit parameters
Initialize learning parameters
For all the stages Identify possible state vectors, χ_k
Evaluate minimum and maximum demands permissible
Initialize $Q^0(x_k, a_k) = 0$
Initialize $p_{x_k}(a_k) = 1/n_k$, $n_k$maximum number of actions possible in A_k
For (n = 0 to max_iteration)
Begin
  P_D=rand ($Dmin_0$, $Dmax_0$), $D_0 = P_D$
  For k = 0 to N − 2
  Do
    State tuple $x_k = (k, D_k)$
    Identify the action space $\mathscr{A}_k$ using Eq. (7)
    Select an action a_k using $p_{x_k}()$
    Apply action $a_k$and find the next state, $D_{k+1} = D_k - a_k$
    Calculate the reward function $g(x_k, a_k, x_{k+1})$ using Eq. (8)
    Update Q^n to Q^{n+1} using Eq. (9)
    Update probability $p^n_{x_k}(a_k)$ to $p^{n+1}_{x_k}(a_k)$ using Eq. (11)
  End do
  $a_{N-1} = D_{N-1}$
  Calculate the reward using Eq. (8)
  Update $Q^n$to Q^{n+1}using Eq. (10)
  Update probability $p^n_{x_k}(a_k)$ to $p^{n+1}_{x_k}(a_k)$ using Eq. (11)
  End.
  Save Q values.

Policy retrieval algorithm:

Read the Q values saved in the learning phase
Initialize P_D = Total power to be dispatched
$D_0 = P_D$
For (k = 0 to N − 1)
  Do
    State tuple $x_k = (k, D_k)$
    Find greedy action $a_g = \text{argmin}_a(Q(x_k, a))$
    Scheduled Power $P_k = a_g$
    $D_{k+1} = D_k - P_k$
  End do

Thus, on executing the learning algorithm and then retrieving the schedule by finding the greedy action corresponding to the input power to each stage of the multi stage decision making task, optimum schedule is obtained for any value of load demand.

Till now, the transmission losses in the system are neglected. Now the Reinforcement Learning approach is extended in order to accommodate the transmission losses occurring in the system.

6. RL algorithm for Economic Dispatch considering transmission losses

The loss in a transmission network can be estimated by executing power flow algorithm or can be approximated using *B*-matrix loss formula. In order to find the schedule accounting the transmission losses, one of the previous algorithms can be used to carry out the learning steps. Being more efficient and faster, pursuit learning strategy is employed for learning the system, generating random values of initial demand. Once the learning phase is completed, the policy retrieval steps provide us with the optimum schedule for any load value. In order to incorporate the transmission losses, the learning is carried out first and policy retrieval is done successively for different values of load values, accounting the losses.

First learn the *Q* value for the different state–action pairs. Schedule for the required load demand is retrieved by policy retrieval phase. For the schedule obtained, transmission losses are calculated by either finding the power flows or using *B*-matrix loss formula. The input demand is then modified by adding the calculated loss MW. The learning algorithm proceeds to find the dispatch for

Table 3
Algorithm for Economic Dispatch considering transmission losses.

Get unit parameters including B-coefficient matrix of the system
Learn the Q values using Pursuit method (Table 2)
Calculate the range of load values possible $D_{min(0)}$ to $D_{max(0)}$
Initial load $P_D = D_{min(0)}$
 Do
  Initialize P_{loss} and Prev_loss to zero
  Initialize final loss tolerance to a small value μ
  Initialize change in loss (Δ) to zero
  Do
    Prev_loss=P_{loss}
    Find the allocation corresponding to P_D using the policy retrieval
    Find the loss using B coefficients as P_{loss}
    Update $P_D = P_D + P_{loss}$
    Compute change in loss $\Delta = P_{loss}$ − prev_loss
  while ($\Delta > \mu$)
  Increment the load P_D with suitable discrete step value.
While ($P_D <= D_{max(0)}$)

the new demand value, giving out the new schedule of generation. This new power allocation will certainly give a new value of transmission loss, which is again used for updating the demand.

The iterative procedure is continued until the loss calculation converges (indicated by the change in loss in two successive iterations coming within tolerable limits). By following these steps iteratively for different load values ranging from $D_{min(0)}$ to $D_{max(0)}$, economic allocation schedule for the entire range of possible demand (with the given generating units) is obtained. Algorithm incorporating transmission loss to find the schedule for all the possible values of load demand is presented in Table 3.

The discrete step for load MW is taken as 10 MW so as to manage the number of states at each stage of the problem. Value of μ is taken as 1 MW so that transmission loss, less than 1 MW can be neglected compared to the load power. Once the learning phase is completed, the economic allocation for all the possible load values can be obtained instantaneously. The main attraction of these algorithms comes from the fact that the learning is carried out only once and need not be repeated for each load demand as in other stochastic methods.

7. Performance of algorithms

The proposed Economic Dispatch algorithms are assessed using different standard test cases. RL based Economic Dispatch can be applied for finding the schedule for generating units when the cost of generation is provided in any of the different forms like variable cost table, cost coefficient, piecewise cost functions and actual cost data from a plant. This becomes useful when the cost of power varies in every block of time since the availability of power is practically a dynamic one.

Algorithms are coded in C language and compiled and executed in GNU Linux environment. Performance evaluation is done with Pentium IV, 2.9 GHz, 512MB RAM personal computer.

In order to validate the proposed algorithms and make a comparison among them, first a three generator system with cost data given in a tabular form is considered [13]. The first two algorithms are executed to find the dispatch without transmission losses and a comparison of execution time is made.

Then IEEE 30 bus system with six generating units having quadratic cost functions [34] is taken in order to prove the efficacy of the proposed approaches. The suitability of the proposed algorithms for a system having generating units with piecewise cost functions is also studied by considering a 10 generator system. The last algorithm is validated and the flexibility of the Reinforcement Learning solution is investigated for system with 20 generating units having given cost functions.

In order to apply Reinforcement Learning algorithms, first the learning parameters are to be fixed based on the problem environment. The learning parameter ε accounts for rate of exploration and exploitation needed. Since it indicates a probability, it can take any positive value less than 1. A small fixed value may result in premature convergence of the learning algorithm while a large fixed value may make the system oscillatory. Therefore in these RL based algorithms, a value of 0.5 is assumed initially providing sufficient exploration of the search space and is decreased by a small factor successively.

Discount parameter γ accounts for the discount to be made in the present state for accounting of future reinforcements and since in the case of Economic Dispatch problem, the cost of future stages has the same implication as the cost of the current stage, value of γ is taken as 1.

The step size of learning is given by the parameter α and it affects the rate of modification of the estimate of Q value at each iteration step. By trial and error α is taken as 0.1 for achieving sufficiently good convergence of the learning system. The RL parameters used in the dispatch problem are tabulated in Table 4

Case I – three generator system

First a simple example with three generating units [13] is considered for validating and explaining the RL approach of solution. The transmission losses are neglected in this case. The cost details are given in tabular form, which can be obtained from experience in case of a practical system. The unit characteristics are given in Table 5, where C_i stands for the cost of generating P MW by ith unit.

The three generating units are having the minimum and maximum power generation possible as (50, 200), (50, 150) and (50, 175). The discretization step for state space and action space, S_s and S_a are taken as 25 MW.

Therefore,

$$Dmin_{(0)} = 150,\quad Dmax_{(0)} = 525$$
$$Dmin_{(1)} = 100,\quad Dmax_{(1)} = 325$$
$$\text{and } Dmin_{(2)} = 50,\quad Dmax_{(2)} = 175$$

In order to understand the performance of RL algorithms, the different components of the multi stage decision process are to be identified. The state tuples is of the form (k, D_k), D_k being the power to be dispatched at kth stage.

Table 4
RL parameters.

ε	*0.5*
α	*0.1*
γ	*1*

Table 5
Cost table for three generator system.

P (MW)	C_1	C_2	C_3
0	100,000	100,000	100,000
25	100,000	100,000	100,000
50	810	750	806
75	1355	1155	1108.5
100	1460	1360	1411
125	1772.5	1655	11704.5
150	2085	1950	1998
175	2427.5	100,000	2358
200	2760	100,000	100,000
225	100,000	100,000	100,000

Table 6
Allocation schedule for three generator system.

D (MW)	$P1$ (MW)	$P2$ (MW)	$P3$ (MW)	Cost (Rs)
250	50	50	150	3558
275	50	150	75	3868.5
300	50	100	150	4168
325	50	125	150	4463
350	50	150	150	4758
375	100	125	150	5113
400	100	150	150	5408
425	125	150	150	5720.5
450	150	150	150	6033
475	175	150	150	6375.5
500	200	150	150	6708

Table 7
Comparison execution time for ε greedy and pursuit solutions.

	ε greedy	Pursuit
No: of iterations	100,000	50,000
Computation time (s)	2.567	1.754

Then, state space $\chi = \chi_0 U \chi_1 U \chi_2$ where

$$\chi_0 = \{(0,150),(0,175),(0,200),\ldots\ldots\ldots\ldots,(0.525)\}$$
$$\chi_1 = \{(1,100),(1,125),(1,150),\ldots\ldots\ldots\ldots,(1,325)\}$$
$$\text{and } \chi_2 = \{(2,50),(2,75),(2,100),\ldots\ldots\ldots\ldots,(2,175)\}$$

Now identify the action space, which is a dynamic one since it depends on the value of power D_k to be dispatched. The minimum and maximum values of actions are found out as

$$Min_0 = (D_0 - D_{max(1)}) \text{ or } P_{min(0)} \text{ whichever is greater}$$
$$Max_0 = (D_0 - D_{min(1)}) \text{ or } P_{max(0)} \text{ whichever is smaller}$$
$$Min_1 = (D_1 - D_{max(1)}) \text{ or } P_{min(1)} \text{ whichever is greater}$$
$$Max_2 = (D_1 - D_{min(1)}) \text{ or } P_{max(1)} \text{ whichever is smaller}$$
$$Min_2 = P_{min(2)},\quad Max_2 = P_{max(2)}$$

For the purpose of explaining the algorithm, let the random value generated for the demand be 300 MW. Then the action space at stage$_0$ is

$$A_0 = \{50, 75, \ldots\ldots\ldots, 200\}$$

One of these actions is selected and it passes to the next stage $k = 1$. Then the action space $\mathscr{A}_1$ is identified and action selection is continued.

The two algorithms, based on ε greedy and pursuit are executed. Both the algorithm gave same results. The allocation schedule for a load demand of 300 is obtained as (50,100,150) and the cost of generation is Rs. 4168/-. Using policy retrieval phase, power schedule for any value of possible input demand values can be retrieved. The two algorithms are run for various values of power demand $D_{min(0)} <= P_D <= D_{max(0)}$, i.e., $150 <= P_D <= 525$. Part of the simulation result is tabulated in Table 6 which is consistent with results given in [13].

For comparing the efficacy of the two algorithms, Simulation time for the two algorithms are compared in Table 7

On comparing the computation time and performance of the algorithms, Algorithm II seems to be better.

Case II – IEEE 30 bus system

To prove the flexibility , the proposed algorithms are now tested for IEEE 30 bus system consisting of six generators [34], without considering the transmission losses. The system cost data is given in quadratic cost coefficient form as given in Table 8. i.e., for any

Table 8
Cost coefficients for IEEE 30 bus system.

Ca	Cb	Cc	$P_{\min}$ (MW)	$P_{\max}$ (MW)
561	7.92	0.001562	150	600
310	7.85	0.00194	100	400
78	7.978	0.00482	50	200
102	5.27	0.00269	100	500
51	9.9	0.00172	40	350
178	8.26	0.00963	100	280

Table 9
Allocation schedule for IEEE 30 bus system.

D (MW)	P_1 (MW)	P_2 (MW)	P_3 (MW)	P_4 (MW)	P_5 (MW)	P_6 (MW)	Cost (Rs)
600	150	100	50	160	40	100	5951.611
700	150	100	50	260	40	100	6591.591
800	150	100	50	360	40	100	7285.371
900	150	100	50	460	40	100	8032.951
1000	160	150	50	500	40	100	8847.839
1100	210	190	60	500	40	100	9698.202
1200	260	220	80	500	40	100	10563.33
1300	310	260	90	500	40	100	11443.07
1400	350	300	110	500	40	100	12337.4
1500	400	340	120	500	40	100	13246.5
1600	440	380	140	500	40	100	14170.28
1700	500	400	160	500	40	100	15109.32
1800	580	400	180	500	40	100	16070.22
1900	600	400	200	500	100	100	17070.12
2000	600	400	200	500	180	120	18108.22
2100	600	400	200	500	270	130	19175.55
2200	600	400	200	500	350	150	20,272
2300	600	400	200	500	350	250	21483.2

power P, cost of generation is given out by the equation $C(P) = Ca + Cb^{*}P + Cc^{*}P^2$, where Ca, Cb and Cc are the cost coefficients.

Also the maximum and minimum generations possible for each of the six generators are specified.

The maximum generation possible with these six generators turns out to be 2330 MW while minimum generation is 540 MW. The RL algorithms are now applied to get the economic allocation for the six units.

The discretization step for action and state space are taken as 10 MW as balance between the accuracy and size of the state and action spaces. At each step of iteration, action is selected according to the exploration method. The Q values of state–action pairs are updated for which cost of generation is calculated by evaluating the quadratic equation.

In case of ε greedy method, after 5×10^5 iterations the Q values approach optimum while pursuit solution converged in 2×10^5 iterations. The optimum dispatch is found out by tracing out the greedy action which give out minimum Q value corresponding to a particular state D_k as k vary from 0 to $N-1$. The optimum schedule for the different values of power demand is obtained using the policy retrieval phase. Schedule for the entire load values ranging from 540 MW to 2330 MW is obtained. The entire schedule is obtained in 23.87 s and 15.63 s respectively for the two algorithms which proves the suitability of the algorithms

The algorithms are executed in several trials and the cost and allocation schedule obtained are almost the same with negligible error in the different trials of the solutions.

A part of the allocation schedule corresponding to various values of power demand in steps of 100MW is tabulated in Table 9.

Case III – 10 Generator system with piece wise quadratic cost functions

To verify the algorithms for non convex cost functions and compare with one of the recent techniques, 10 generator system having piecewise quadratic cost functions [24] is considered. The different generators are having two or three different operating regions. If the Cost function is C_i and the space interval is divided into three divisions, then it is represented as follows:

$$\begin{aligned} C_i(P) &= a_{i(1)} + b_{i(1)}P_i + c_{i(1)}P_i^2 (P_{min(i)} \leqslant P_i \geqslant P_{i(1)}) \\ &= a_{i(2)} + b_{i(2)}P_i + c_{i(2)}P_i^2 (P_{i(1)} \leqslant P_i \geqslant P_{i(2)}) \\ &= a_{i(3)} + b_{i(3)}P_i + c_{i(3)}P_i^2 (P_{i(2)} \leqslant P_i \geqslant P_{max(i)}) \end{aligned}$$

The data $(a_i, b_i, c_i, P_{min}, P_{max})$ of generators are given in Table 10

Table 10
Generator data for 10 generator system.

Gen.	$P_{\min}$ (MW)	$P_{\max}$ (MW)	a	b	c
1	100	196	26.97	−0.3975	0.002176
1	196	250	21.13	−0.3059	0.001861
2	50	114	1.865	−0.03988	0.001138
2	114	157	13.65	−0.198	0.00162
2	157	230	118.4	−1.269	0.004194
3	200	332	39.79	−0.3116	0.001457
3	332	388	−2.876	0.03389	0.000804
3	388	500	−59.14	0.4864	1.18E−05
4	99	138	1.983	−0.03114	0.001049
4	138	200	52.85	−0.6348	0.002758
4	200	265	266.8	−2.338	0.005935
5	190	338	13.92	−0.08733	0.001066
5	338	407	99.76	−0.5206	0.001597
5	407	490	53.99	0.4462	0.00015
6	85	138	1.983	−0.03114	0.001049
6	138	200	52.85	−0.6348	0.002758
6	200	265	266.8	−2.338	0.005935
7	200	331	18.93	−0.1325	0.001107
7	331	391	43.77	−0.2267	0.001165
7	391	500	43.35	0.3559	0.000245
8	99	138	1.983	−0.03114	0.001049
8	138	200	52.85	−0.6348	0.002758
8	200	265	266.8	−2.338	0.005935
9	130	213	14.23	−0.01817	0.000612
9	213	370	88.53	−0.5675	0.001554
10	362	407	46.71	−0.2024	0.001137
10	407	490	61.13	0.5084	4.16E−05
10	407	490	61.13	0.5084	4.16E−05

The system is made to learn using the two algorithms given in Tables 1 and 2 and the Q values approach optimum in 5×10^6 and 1.5×10^7 iterations respectively. The same values of learning parameters are taken as in previous cases. The discretization step for state and action spaces is taken as 10 MW.

Part of the allocation schedule corresponding to values of power demand ranging from 1400 MW to 3000 MW obtained are given in Table 11. The times of execution are 38.69 s and 34.95 s respectively. The cost and allocation schedule obtained are comparable with that of improved genetic algorithm [24].

The simulation results proved the suitability and efficiency of the proposed algorithms for scheduling of generators with different categories of cost functions. Now incorporating the transmission losses, schedule for 20 generating unit system is obtained.

Case IV – 20 Generator system

For validating the Reinforcement Learning algorithm accounting the transmission losses occurring in the system and to prove the scalability of Reinforcement Learning based algorithms, next a 20 generator system is considered. The cost function is given in quadratic form. The unit details are given in Table 12.

The transmission loss is calculated using B coefficient matrix. Algorithm given in Table 3 is executed to give the schedule for the range of load from 1000 MW to 3000 MW. The execution time taken is 37.97 s Part of the schedule and the loss are tabulated in Table 13.

Table 11
Part of schedule – 10 generator system.

Demand (MW)	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	Cost (Rs)
1400	100	170	200	190	190	150	50	200	50	100	903.30
1800	100	220	200	260	270	150	250	200	50	100	1101.472
2200	120	220	210	430	450	150	250	220	50	100	1411.387
2700	210	220	350	500	500	150	250	350	50	120	1960.531
3000	250	220	440	500	500	170	250	440	80	150	2399.784

Table 12
Generator details of 20 generator system.

Unit	C_a	C_b	C_c	$P_{\min}$ (MW)	$P_{\max}$ (MW)
1	1000	18.19	0.00068	150	600
2	970	19.26	0.00071	50	200
3	600	19.8	0.0065	50	200
4	700	19.1	0.005	50	200
5	420	18.1	0.00738	50	160
6	360	19.26	0.00612	20	100
7	490	17.14	0.0079	25	125
8	660	18.92	0.00813	50	150
9	765	18.97	0.00522	50	200
10	770	18.92	0.00573	30	150
11	800	16.69	0.0048	100	300
12	970	16.76	0.0031	150	500
13	900	17.36	0.0085	40	160
14	700	18.7	0.00511	20	130
15	450	18.7	0.00398	25	185
16	370	14.26	0.0712	20	80
17	480	19.14	0.0089	30	85
18	680	18.92	0.00713	30	120
19	700	18.47	0.00622	40	120
20	850	19.79	0.00773	30	100

Table 13
Schedule for 20 generator system.

D (MW)	2000	2500	3000
P_1	421	498	530
P_2	140	159	167
P_3	105	120	140
P_4	94	118	135
P_5	81	92	97
P_6	51	74	87
P_7	89	115	141
P_8	81	106	162
P_9	84	103	155
P_{10}	70	98	108
P_{11}	236	290	358
P_{12}	89	120	136
P_{13}	82	119	124
P_{14}	90	115	147
P_{15}	23	30	70
P_{17}	64	87	111
P_{19}	72	100	106
P_{20}	45	54	75
Loss	39	64	81

Table 14
Comparison of computation times of different RL algorithms (time in seconds).

	ε greedy	Pursuit	Pursuit with transmission losses
Three gen system with given cost table, neglecting transmission loss	2.567	1.754	
IEEE 30 bus system with 6 generators, neglecting transmission losses	23.87	15.63	
10 generator system with piecewise quadratic cost coefficients	42.87	38.69	
20 Gen. system considering transmission losses			37.97

The different Reinforcement Learning algorithms have been tested for their efficacy and performance. The computation time of the different RL algorithms for the different test cases are tabulated for comparison in Table 14.

8. Conclusion

In this paper, two RL based algorithms to solve Economic Dispatch problem was presented. The algorithms were validated using different test cases found in literature, and the performance were found to be promising. In the proposed solution strategy, with a single learning task the schedule for any load demand can be easily retrieved. This makes the algorithm efficient compared to other soft computing techniques. Also in the reinforcement Learning solution, the reward function which in this case is the cost of generation need not be a mathematically defined function. It can take stochastic cost data from a real time environment. In other words the algorithm is suitable to accommodate any cost characteristics.

RL approach to Economic Dispatch is a promising area of research work. With further work , RL based algorithms can be used to learn optimum schedule from real time data. Reinforcement Learning approach is a powerful tool for solving optimisation problem. Developments in RL could be exploited to develop better algorithms for various optimization problems in power systems.

Acknowledgment

The second author gratefully acknowledges the support of All India Council of Technical Education under the TAPTEC scheme (Ref. No. 8021/RID/NPROJ/TAP-139/2002-03).

References

[1] Bertsekas DP, Tsitsikilis JN. Neurodynamic programming. Belmount, MA: Athena Scientific; 1996.
[2] Sutton RS, Barto AG. Reinforcement learning: an introduction. Cambridge, MA: MIT Press; 1998.
[3] Anderson CW. Learning to control an inverted pendulum using Neural Networks. IEEE Control Syst Mag 1989;9:31–7.
[4] Tasauro GJ, gammon TD. A self teaching backgammon program achieves master level play. Neural Comput 1994;6:215–9.
[5] Crytes Robort H, Barto Andrew G. Elevator group control using multiple reinforcement learning agents. Boston: Kulwer Academic; 2002.
[6] Imthias Ahamed TP, NagendraRao PS, Sastry PS. A reinforcement learning approach to automatic generation control. Electr Power Syst Res 2002;63:9–26.
[7] Imthias Ahamed TP. A reinforcement learning approach to unit commitment problem. In: Proceedings of national power system conference; 2006.
[8] Galvic M, Ernst D, Wehenkel L. A reinforcement learning based discrete supplementary control for power system transient stability enhancement. In: Proceedings on intelligent system application to power system, Vollos, Greece; 2003.
[9] Nanduri Vishnuteja, Das Tapas K. A reinforcement learning model to assess market power under auction based energy pricing. IEEE Trans Power Syst 2007;22(1):85–95.
[10] Gajjar GR, Khaparde SA, Nagaraju P, Soman SA. Application of Actor Critic Learning algorithm for optimal bidding problem of a Genco. IEEE Trans Power Syst 2003;18(1):11–8.
[11] Jasmin EA, Imthias Ahamed TP, Jagathiraj VP. A Reinforcement Learning algorithm to Economic Dispatch considering transmission losses. In: Proceedings of TENCON 2008 IEEE Region 10 conference; 2008. p. 1–6.

[12] Jasmin EA, Imthias Ahamed TP, Jagathiraj VP. A Reinforcement Learning approach to Economic Dispatch using Neural Networks. In: Proceedings of national power system conference. Bombay: IIT; 2008.
[13] Wood AJ, Woolenberg BF. Power generation and control. John Wiley Sons; 2002.
[14] Chowdhury BH, Rahman Salfur. A review of recent advances in economic dispatch. IEEE Trans Power Syst 1990;5(4):1248–59.
[15] Senthilkumar S, Palnisamy V. A Dynamic Programming based fast computation Hopfield network for Unit Commitment and Economic Dispatch. Electr Power Syst Res 2006.
[16] Balakrishnan S, Kannan PS, Aravindan C, Subathra P. On line emission and economic load dispatch using Hopfield neural network. Appl Soft Comput 2003;2:297–305.
[17] Su Ching Tzong, Lin Chien Tung. New approach with a Hopfield modeling framework to economic dispatch. IEEE Trans Power Syst 2000;15(2):541–5.
[18] Xia Quing, Song YH. Effective decomposition and co ordination algorithms for Unit Commitment and Economic Dispatch with security constraints. Electr Power Syst Res 2000;53:71–80.
[19] Ji PyngChiou. Variable scaling hybrid differential evolution for large scale Economic Dispatch Problems. Electr Power Energy Syst 2007;77:212–8.
[20] Penigrahi BK, Yadav Salik R, Agarwal Shubham. A clonal algorithm to solve economic load dispatch. Electr Power Syst Res 2006.
[21] Aravindababu P, Nayer KR. Economic dispatch based on optimal lambda using radial basis function network. Electr Power Energy Syst 2005;21:551–6.
[22] Nanda J, Badrinarayanan R. Application of genetic algorithm to economic load dispatch with line flow constraints. Electr Power Energy Syst 2001;10:243–51.
[23] Basu M. A simulated annealing based goal attainment method for economic emission load dispatch of fixed head hydro thermal power systems. Electr Power Energy Syst 2004;24:147–53.
[24] Won Jong Ryul, Park Young Moon. Economic dispatch solutions with piecewise quadratic functions using improved genetic algorithm. Electr Power Energy Syst 2003;25:355–61.
[25] Chen Chung Lung, Chen Nanming. Direct search method for solving economic dispatch problem considering transmission capacity constraints. IEEE Trans Power Syst 2001;16(1):764–9.
[26] Chen Chun Lung. Non convex economic dispatch: a direct search approach. Energy Convers Manage 2007;48:219–25.
[27] Jayabharthi T, Jayprakash K. Evolutionary programming techniques for different kinds of economic dispatch problems. Electr Power Syst Res 2005;73:169–76.
[28] Somasundaram P, Kuppusamy K. Application of evolutionary programming to security constrained economic dispatch. Electr Power Energy Syst 2005;27:343–51.
[29] Immanuel Selvakumar A, Thaushikodi K. A New particle swarm optimization solution to non convex economic dispatch problems. IEEE Trans Power Syst 2007;22(2):42–51.
[30] Wang Lingfeng, Singh Chanan. Environmental economic power dispatch using fuzzified multi objective particle swarm optimization algorithm. Electr Power Syst Res 2007.
[31] Liu Derong, Cai Ying. Taguchi method for solving economic dispatch problem with non smooth fuel cost functions. IEEE Trans Power Syst 2005;4(2):2006–14.
[32] Thalchar MA, Sastry PS. Networks of learning automata: techniques for online stochastic approximation. Boston: Kluwer Academic; 2003.
[33] Si Jennie, Barto Andy, Powell Warren, Wunsch Donald. Hand book of learning and approximate dynamic programming. IEEE Press, John Wiley & Sons; 2004.
[34] Pai MA. Computer techniques in power system analysis. New Delhi: Tata McGraw Hill; 1979.