2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm) | 978-1-6654-1502-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/SMARTGRIDCOMM51999.2021.9632339

Solving Unit Commitment Problems with Multi-step Deep Reinforcement Learning

Jingtao Qin, Nanpeng Yu and Yuanqi Gao
Department of Electrical and Computer Engineering
University of California, Riverside
Riverside, California 92507 USA
jqin020@ucr.edu, nyu@ece.ucr.edu, ygao024@ucr.edu

***Abstract*—Solving the unit commitment (UC) problem in a computationally efficient manner is a critical issue of electricity market operations. Optimization-based methods such as heuristics, dynamic programming, and mixed-integer quadratic programming (MIQP) often yield good solutions to the UC problem. However, the computation time of optimization-based methods grows exponentially with the number of generating units, which is a major bottleneck in practice. To address this issue, we formulate the UC problem as a Markov decision process and propose a novel multi-step deep reinforcement learning (RL)-based algorithm to solve the problem. We approximate the action-value function with neural networks and design an algorithm to determine the feasible action space. Numerical studies on a 5-generator test case show that our proposed algorithm significantly outperforms the deep Q-learning and yields similar level of performance as that of MIQP-based optimization in terms of optimality. The computation time of our proposed algorithm is much shorter than that of MIQP-based optimization methods.**

***Index Terms*—Unit commitment, Markov decision process, deep reinforcement learning, multi-step return.**

I. Introduction

The unit commitment (UC) problem in the day-ahead market determines the optimal startup and shutdown schedules of generators based on supply offers, demand bids, network conditions and operational constraints. Achieving near-optimal UC solutions is vital to improving the efficiency of day-ahead market. The UC problem is often first formulated as Mixed Integer Quadratic Programming (MIQP) problem and then solved by commercially available solvers. The existing literature on UC problems can be categorized into two groups according to the model assumption.

Methods in the first group leverage model-based methods including heuristic methods [1], dynamic programming [2], mixed-integer linear programming [3], Lagranian relaxation [4], and meta-heuristics such as simulated annealing [5] and evolution approaches [6]. Although these methods produce good results, their computation time increase exponentially with the number of energy resources and operational constraints. As the number of aggregated distributed energy resources and uncertainties associated with renewable energy continue to increase, it will be challenging to find a near optimal UC solution in a computationally efficient manner.

In the second group of literature, reinforcement learning (RL)-based methods have been used to solve the UC problem. RL is a mathematical framework for learning to solve sequential decision-making problems. It has been used to solve control problems in power distribution systems [7]. [8] proposes three reinforcement learning techniques, which include approximate policy iteration, tree search, and back sweep to reduce system operation costs compared to simulated annealing on a 12-units system. In [9], Navin N.K et al. cast the UC problem as a multi-agent fuzzy reinforcement learning task where individual generators act as players to jointly minimize the total operational cost. A decentralized Q-learning-based optimization algorithm is proposed in [10] to solve economic dispatch and UC problems in an online manner.

The RL-based algorithms mentioned above face a challenge that the dimensionality of state and action space increases exponentially with the size of the problem. To address this issue, we propose a multi-step deep reinforcement learning algorithm for UC problems, which uses deep Q-learning network to parameterize an approximate action-value function. To the best of our knowledge, this work is the first attempt to apply deep reinforcement learning to solve UC problems. The reminder of this paper is organized as follows: Section II presents the problem formulation of UC problems. Section III provides the technical methods. Section IV shows the results of numerical studies and the robustness of proposed algorithm. Section V states the conclusion.

II. Problem Formulation

In this section, we first describe the formulation of the UC problem. Then we review the basics of the Markov decision process (MDP). Finally, we formulate the UC problem as an MDP.

A. Formulation of Unit Commitment Problems

The objective of the unit commitment problem is to find the optimal unit schedule that minimizes the total operation cost over the operational horizon as shown in (1).

$$\text{Min} \sum_{t=1}^{T} \sum_{i=1}^{N} c_i^p(t) + c_i^u(t) + c_i^d(t), \tag{1}$$

978-1-6654-1502-6/21/$31.00 ©2021 IEEE

where T is the number of time periods in the operational horizon, N is the number of units, $c_i^p(t)$ is the production cost of unit i in period t, $c_i^u(t)$ is the startup cost of unit i in period t, and $c_i^d(t)$ is the shutdown cost of unit i in period t.

In this paper, we consider four types of constraints, namely, the load/spinning reserve balance constraints, generation limits, ramping limits, and minimum up and down time constraints.

1) Load/Spinning Reserve Balance Constraints: The energy balance constraints and spinning reserve requirements are enforced by (2)-(3).

$$\sum_{i=1}^{N} p_i(t) = d(t),\ \forall t \in T \tag{2}$$

$$\sum_{i=1}^{N} \bar{p}_i \geq d(t) + R(t),\ \forall t \in T, \tag{3}$$

where $\bar{p}_i(t)$ is the maximum available power output of unit i in period t, $d(t)$ and $R(t)$ are the load demand and spinning reserve requirement in period t. $p_i(t) \in \prod_i(t)$ is the power output of unit i in period t, $\prod_i(t)$ is the feasible production region of unit i in period t, which is determined by the operation constraints as follows.

2) Generation Limits: The generation units' operating constraints are enforced by (4)-(5):

$$\underline{P}_i v_i(t) \leq p_i(t) \leq \bar{p}_i,\ \forall i \in N,\ \forall t \in T \tag{4}$$

$$0 \leq \bar{p}_i(t) \leq \bar{P}_i v_i(t),\ \forall i \in N,\ \forall t \in T, \tag{5}$$

where $\bar{P}_i$ is the capacity of unit i. $\underline{P}_i$ is the minimum output of unit i. $v_i(t)$ is the on/off status of unit i in period t.

3) Ramping Limits: The output of generation units are also constrained by ramp up and startup ramp rates (6), shutdown ramp rates (7), as well as ramp down limits (8):

$$\begin{aligned}\bar{p}_i(t) \leq & p_i(t-1) + \mathrm{RU}_i v_i(t-1) \\ & + \mathrm{SU}_i[v_i(t) - v_i(t-1)] + \bar{P}_i(1 - v_i(t))\end{aligned} \tag{6}$$

$$\begin{aligned}\bar{p}_i(t) \leq & \bar{P}_i v_i(t+1) \\ & + \mathrm{SD}_i[v_i(t) - v_i(t+1)]\end{aligned} \tag{7}$$

$$\begin{aligned}p_i(t-1) \leq & p_i(t) + \mathrm{RD}_i v_i(t) \\ & \mathrm{SD}_i[v_i(t-1) - v_i(t)] + \bar{P}_i[1 - v_i(t-1)],\end{aligned} \tag{8}$$

where RU_i and RD_i are the ramp up and down limits of unit i. SU_i and SD_i are the startup and shutdown ramp limits of unit i.

4) Minimum Up and Down Time: The minimum up and down time constraints can be formulated as mixed-integer linear equations in (9)-(14):

$$\sum_{t=1}^{G_i}[1 - v_i(t)] = 0, \forall i \in N \tag{9}$$

$$\begin{aligned}\sum_{n=t}^{t+\mathrm{UT}_i-1} v_i(n) \geq \mathrm{UT}_i[v_i(t) - v_i(t-1)], \\ \forall i \in N,\ \forall t = G_i + 1, \cdots, T - \mathrm{UT}_i + 1\end{aligned} \tag{10}$$

$$\begin{aligned}\sum_{n=t}^{T}\{v_i(n) - [v_i(t) - v_i(t-1)]\} \geq 0 \\ \forall i \in N,\ \forall t = T - \mathrm{UT}_i + 2, \cdots, T\end{aligned} \tag{11}$$

$$\sum_{t=1}^{L_i}[v_i(k)] = 0, \forall i \in N \tag{12}$$

$$\begin{aligned}\sum_{n=t}^{t+\mathrm{DT}_i-1} [1 - v_i(n)] \geq \mathrm{DT}_i[v_i(t-1) - v_i(t)], \\ \forall i \in N,\ \forall t = L_i + 1, \cdots, T - \mathrm{DT}_i + 1\end{aligned} \tag{13}$$

$$\begin{aligned}\sum_{n=t}^{T}\{1 - v_i(n) - [v_i(t-1) - v_i(t)]\} \geq 0 \\ \forall i \in N,\ \forall t = T - \mathrm{DT}_i + 2, \cdots, T\end{aligned} \tag{14}$$

where G_i and L_i are the numbers of initial periods during which unit i must be online or offline. UT_i and DT_i are the minimum up and down time of unit i.

B. Basics of Markov Decision Process (MDP)

MDP is the mathematical framework for describing sequential decision making problems. It can be formalized as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the environment transition probability, $\mathcal{R}$ is the reward function and γ is a discount factor ($0 \leq \gamma \leq 1$) [11]. At each time step t, the agent selects an action a_t from the action space $\mathcal{A}$ based on the current state $s_t \in \mathcal{S}$. Then it receives a numerical reward $r_{t+1} = \mathcal{R}(s_t, a_t)$ and the environment transitions to the next state s_{t+1} according to the transition probability $\mathcal{P}(s_{t+1}|s_t, a_t)$.

The goal of the agent is to learn a policy $\pi(a|s)$ that maximizes the total expected discounted rewards $J(\pi) = \mathbf{E}[G(\tau)]$, where T is the length of one episode, $G(\tau) = \sum_{t=0}^{T} \gamma^t r_{t+1}$, and τ is a trajectory of states and actions $s_0, a_0, r_1, s_1, a_1, r_2, ...$ Here we need to define two important value functions $v_\pi(s)$ and $q_\pi(s, a)$ to represent the value of states and state-action pairs following a policy π:

$$\begin{aligned}v_\pi(s) &= \mathbf{E}_\pi[G_t|S_t = s] \\ &= \mathbf{E}_\pi\left[\textstyle\sum_{k=0}^{T} \gamma^k r_{t+k+1}|S_t = s\right]\end{aligned} \tag{15}$$

$$\begin{aligned}q_\pi(s, a) &= \mathbf{E}_\pi[G_t|S_t = s, A_t = a] \\ &= \mathbf{E}_\pi\left[\textstyle\sum_{k=0}^{T} \gamma^k r_{t+k+1}|S_t = s, A_t = a\right]\end{aligned} \tag{16}$$

The optimal policy is defined as $\pi(a|s) = \arg\max_\pi v_\pi(s)$ for all $s \in S$ or $\pi(a|s) = \arg\max_\pi q_\pi(s, a)$ for all $s \in S$ and $a \in A(s)$.

C. Formulate the UC Problem as an MDP

In this subsection, we formulate the UC problem as an MDP. The overall schematic of the MDP formulation for the UC problem is shown in Figure 1. The episode, state, action, and reward function are defined as follows.

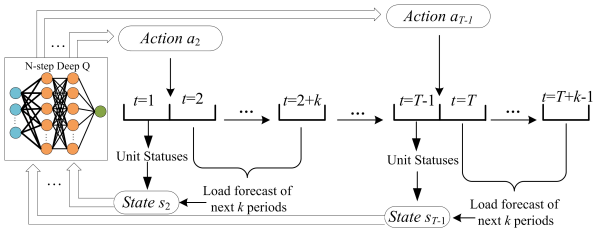

Figure 1: MDP formulation of the UC problem.

Algorithm 1 Compute Feasible Actions Set

Initialize feasible action set $\mathcal{A}_t = \varnothing$, from current state $\boldsymbol{s}_t$ obtain $\boldsymbol{v}_t, \boldsymbol{p}_t, \boldsymbol{u}_t, \boldsymbol{d}_t$

for $n = 1, \cdots, 2^N$ **do**
  Obtain action vector $\boldsymbol{v}_{t+1}^n$
  set $\nu_1 = \nu_2 = \nu_3 = 1$
  set vector $\bar{\boldsymbol{p}}$ to zero
  for $i = 1, \cdots, N$ **do**
    if $u_i(t) < v_i(t)\mathrm{UT}_i + (1 - v_i)\mathrm{DT}_i$ **then**
      if $v_i^n(t+1) \neq v_i(t)$ **then**
        $\nu_1 = 0$
        break
    if $v_i(t) = 1$ **then**
      if $p_i(t) > \mathrm{SD}_i$ **then**
        if $v_i^n(t+1)$=0 **then**
          ν_2=0
          break
    if $v_i^n(t+1) = 1$ **then**
      if $v_i(t) = 1$ **then**
        $\bar{p}_i = \min(p_i(t) + \mathrm{RU}_i, \bar{P}_i)$
      else $\bar{p}_i = \min(p_i(t) + \mathrm{SU}_i, \bar{P}_i)$
  if $\sum_{i=0}^{N} \bar{p} < d(t+1) + R(t+1)$ **then**
    $\nu_3 = 0$
  if $\nu_1 = \nu_2 = \nu_3 = 1$ **then**
    append $\boldsymbol{v}_{t+1}^n$ to set $\mathcal{A}_t$

1) Time Steps and Episodes: We define each hour as a time step. Here we focus on the day-ahead unit commitment problem. Therefore we define an episode to be either a whole day (24 time steps), or last until no feasible action can be found, whichever comes the first. When an episode is finished, the next episode always starts from the first hour of the next day. We denote the length of an episode as T.

2) States: We define the state at time t as $s_t = (t, \boldsymbol{v}_t, \boldsymbol{p}_t, \boldsymbol{u}_t, \boldsymbol{d}_t)$, where t is the global time, $\boldsymbol{v}_t$ is a vector of the on/off status $v_i(t)$ of unit i in period t (1 if it is on, 0 otherwise), $\boldsymbol{p}_t$ is a vector of the power output $p_i(t)$ of unit i in period t, $\boldsymbol{u}_t$ is a vector of the number of periods $u_i(t)$ that unit i has been on/off up to period t and can be formulated as (17). Note that $v_i(0)$ is the initial commitment state of unit i and $u_i(0)$ is the number of periods that unit i has been on/off prior to the first period of the episode.

$$u_i(t) = \begin{cases} u_i(t-1) + 1, & \text{if } v_i(t) - v_i(t-1) = 0 \\ 1, & \text{otherwise} \end{cases} \tag{17}$$

Finally, $\boldsymbol{d}_t$ is a vector $[d(t+1), d(t+2), \cdots, d(t+K)]$ of load demand forecasts for the next K steps.

3) Actions: The action a_t at time t is defined as changing the on/off status of all units to $\boldsymbol{v}_{t+1}$ in period $t+1$. Due to the operation constraints of generation units and the varying load demand, some of the on/off status might be infeasible. Therefore, we need to find all the feasible actions based on the current state to determine the action space. The algorithm for generating all feasible actions is presented in Algorithm 1. First, we initialize the feasible action set $\mathcal{A}_t = \varnothing$ and obtain $\boldsymbol{v}_t, \boldsymbol{p}_t, \boldsymbol{u}_t, \boldsymbol{d}_t$ from state s_t in period t. Then we check whether all operation constraints are satisfied for each of the 2^N possible combinations. In the algorithm, ν_1, ν_2, ν_3 are the flag variables denoting the satisfaction of minimum up/down time limits, shutdown ramp limits, and spinning reserve requirement, respectively. If all three constraints are satisfied, then $\boldsymbol{v}_{t+1}^n$ is a feasible action.

4) Reward: The reward r_{t+1} reflects the negative of the operation cost C_{t+1} in period $t+1$, which is defined as (18).

$$C_{t+1} = \sum_{i=1}^{N} c_i^p(t+1) + \sum_{i=1}^{N} c_i^u(t+1) + \sum_{i=1}^{N} c_i^d(t+1) \tag{18}$$

- $c_i^p(t+1)$ is the quadratic production cost function in time period $t+1$ given by (19) [12].

$$c_i^p(t+1) = a_i v_i(t+1) + b_i p_i(t+1) + c_i p_i^2(t+1) \tag{19}$$

We can obtain $p_i(t+1)$ by solving a single period economic dispatch (ED) using quadratic programming once the on/off status $v_i(t+1)$ are known. Note that in the single period ED process, the ramp up/down limit and startup ramp limit need to be considered.

- The startup cost $c_i^u(t+1)$ is calculated by a staircase function [3] as in (20).

$$c_i^u(t+1) = \begin{cases} \mathbf{CU}_i\left[\min\{\mathbf{ND}_i, u_i(t)\}\right], & \text{if } v_i(t+1) > v_i(t) \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

where $\mathbf{CU}_i$ is a vector of the staircase startup cost of unit i. The symbol $\mathbf{CU}_i[k]$ denotes taking the k-th element from the vector $\mathbf{CU}_i$. $\mathbf{ND}_i$ is the number of intervals of the staircase startup cost function, i.e. the length of $\mathbf{CU}_i$.

- The shutdown cost $c_i^d(t+1)$ is formulated as (21).

$$c_i^d(t+1) = \begin{cases} \mathbf{CD}_i, & \text{if } v_i(t) > v_i(t+1) \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

where $\mathbf{CD}_i$ is the shutdown cost of unit i.

As mentioned before, an episode could be terminated early if there is no feasible action under the current state. To avoid this situation, we should give the agent a large punishment

when it occurs. To this end, we define the reward function as (22).

$$r_{t+1} = -\begin{cases} C_{t+1}, & \text{if } \mathcal{A}_{t+1} \neq \varnothing \\ \zeta, & \text{if } \mathcal{A}_{t+1} = \varnothing \end{cases} \tag{22}$$

where ζ is a large constant.

III. Technical Methods

In this section, we introduce the proposed multi-step deep reinforcement learning algorithm for the UC problem formulated in Section II. First, we review the preliminary of deep Q-learning (DQN). Then we introduce a multi-step deep Q-learning algorithm for UC problems to improve the learning efficiency of the basic DQN. Lastly, we present a few additional implementation details.

A. Deep Q-Learning Method

Deep Q-learning is an RL algorithm that combines the vanilla Q-learning with deep neural networks in order to solve MDPs with continuous state space [13]. Deep Q-learning approximates the action-value function using a deep neural network called deep Q network (DQN) $Q(s_t, a_t|\theta)$. The DQN is trained to minimize the mean-squared temporal difference error $L(\theta)$ using the stochastic gradient descent:

$$L(\theta) = \mathbf{E}_{(s,a,r,s')\sim\mathcal{D}}\left(r + \gamma \max_{a'} Q(s', a'|\theta') - Q(s, a|\theta)\right)^2, \tag{23}$$

where $Q(s', a'|\theta')$ is another neural network with the identical architecture as $Q(s_t, a_t|\theta)$, called the target network. The parameters θ' of the target network is updated periodically from the Q network weights θ to stabilize the training process. $\mathcal{D}$ is the replay buffer which stores the transition tuples (s, a, r, s').

B. Multi-Step Deep Q-Learning for UC Problems

The deep Q-learning method discussed above is easy to implement and can be directly applied to solve high-dimensional state space MDPs. However, it can be inefficient when applied to the UC problem introduced in Section II. This is because the effects of taking an action may not be immediately reflected in the next reward and could impact the rewards of multiple time steps later. As a result, many updates are required to propagate the reward to the relevant preceding states and actions [14]. This makes the learning process slow and extremely sample-inefficient.

We address the problem by adopting the multi-step return method [15], where the action value function $Q(s_t, a_t|\theta)$ is updated toward a n-step return $R(t)$:

$$L(\theta) = (Q(s_t, a_t|\theta) - R(t))^2, \tag{24}$$

where $R(t)$ is defined as:

$$R(t) = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \max_{a'\in\mathcal{A}'} \gamma^n Q(s_{t+n}, a'|\theta') \tag{25}$$

To compute $R(t)$, we run the agent-environment interaction for n steps to obtain the rewards $r_{t+k}, k = 1, ..., n$ and the n-step next state s_{t+n}, then evaluate (25).

With the n-step target, both the long-term and the short-term effects of an action can be learned by regressing toward the exact reward, rather than by bootstrapping from the target network $Q(s, a|\theta')$. Thus, multi-step deep Q-learning algorithm significantly improves the learning efficiency for the UC problem.

The pseudocode of a multi-step deep Q-learning for UC problems is summarized in Algorithm 2.

Algorithm 2 Multi-Step Deep Q-learning for UC Problems

Initialize parameters of UC problems
Input historical load data set of N_d days
Initialize day $d = 1$
Initialize learning counter $m = 0$
Initialize action-value function Q with random parameters θ
Initialize target network parameters $\theta' = \theta$
Initialize n-step buffer $\mathcal{D}$ as a queue with a maximum length of n

for episode $= 1, \cdots, M$ **do**
  Input historical load data of day d
  Obtain initial state s_1 of day d
  for $t = 1, \cdots, T$ **do**
    Obtain feasible action set $\mathcal{A}_t$ of state s_t according to Algorithm 1.
    With ϵ select a random action a_t from $\mathcal{A}_t$; otherwise select $a_t = \max_{a\in\mathcal{A}_t} Q(s_t, a|\theta')$.
    Obtain the schedule of units on next period $t+1$ based on action a_t.
    Solve a single period ED and calculate reward r_{t+1} according to (18)-(22).
    Calculate $\boldsymbol{u}_{t+1}$ according to (17) and then formulate the next state s_{t+1}.
    Call Algorithm 1 to calculate $\mathcal{A}_{t+1}$.
    if $\mathcal{A}_{t+1} = \varnothing$ **then**
      $\text{done}_t = 1$
    else $\text{done}_t = 0$
    Store $(s_t, a_t, r_{t+1}, s_{t+1}, \mathcal{A}_{t+1}, \text{done}_t)$ in $\mathcal{D}$
    if length$(\mathcal{D}) = n$ or $\text{done}_t = 1$ **then**

$$R = \begin{cases} 0, & \text{done}_t = 1 \\ \max_a Q(s_{t+1}, a|\theta'), & \text{done}_t = 0 \end{cases}$$

      for $i = t, t-1, \cdots, t - \text{length}(\mathcal{D})$, **do**
        $R = r_i + \gamma R$
        Perform a gradient descent step on $(R - Q(s_i, a_i|\theta))^2$
      $m = m + 1$
      if $\text{mod}(m, I_{target}) = 0$ **then**
        Update $\theta' = \theta$
  if day d is over **then**
    $d = \text{mod}(d+1, N_d)$

C. Algorithm Implementation Details

This subsection provides the implementation details of the proposed multi-step deep Q-learning algorithm for UC problems.

• Q-network structure: The Q-networks are standard feed-forward neural networks with state-action pairs as the input and the corresponding Q value as the single output. This Q-network architecture scales linearly with the number of units.

• Episode initialization: In our problem formulation, the goal is to maximize the total reward over all the training days. Thus the initial status of the next day is obtained from the last time period of the current day. To this end, we use the historical load data of the next day for training only when the agent finds a policy that satisfy the load demand of every time period of the current day.

• Time variable encoding: In this work, we only encode the hour-of-day part of the global time step t, which ranges from 0 to 23. t is encoded in two coordinates $[cos(2\pi t/24), sin(2\pi t/24)]$ to reflect its periodic nature [16].

IV. Numerical Studies

In this section, the proposed RL-based algorithm is applied to solve the UC problem for a five-unit system adopted from [6]. We first provide the numerical study setup, then we show the performance and robustness of the proposed algorithm.

A. Experimental Data and Algorithm Setup

The proposed algorithm is applied to solve a 24-hour horizon scheduling problem for a five-unit system [6]. The parameters of five thermal units are shown in Table I. The minimum and maximum of staircase startup cost **CU** of unit i are equal to its hot start cost (hc) and cold start cost (cc). The number of intervals of staircase startup cost function **ND** is equal to cold start hours (ch) plus 1. Initial status is the number of hours each unit has been online (+) or offline (-) prior to the first period of the first day.

Table I: Parameters for the 5-unit system

	Unit 1	**Unit 2**	**Unit 3**	**Unit 4**	**Unit 5**
$\bar{P}$ (MW)	455	130	130	80	55
$\underline{P}$ (MW)	150	20	20	20	55
a ($/h)	1000	700	680	370	660
b ($/MWh)	16.19	16.60	16.50	22.26	25.92
c ($/MW^{2}h)	0.00048	0.002	0.00211	0.00712	0.00413
UT (h)	8	5	5	3	1
DT (h)	8	5	5	3	1
RU (MW)	300	85	85	55	55
RD (MW)	300	85	85	55	55
SU (MW)	300	85	85	55	55
SD (MW)	300	85	85	55	55
hc ($)	4500	550	560	170	30
cc ($)	9000	1100	1120	340	60
ch (h)	5	4	4	2	0
Initial status (h)	8	-5	-5	-3	-1

The load demand of the 225-bus Western Electricity Coordinating Council (WECC) system developed in [17] is adopted and scaled so that it can be served by the five generation units. The historical data from [17] has 184 days of load demand from May 1, 2004 to October 31, 2004. We split it into training, validation and testing datasets. The training dataset has 153 days of load demand from May 1 to September 31. The validation dataset has 14 days of load demand from October 1 to October 14 and the testing dataset has 14 days of load demand from October 15 to October 28.

The hyperparameters of the multi-step deep Q-learning algorithm are provided in Table II, which are tuned individually to reach their best performance. The hyperparameters of DQN is the same as multi-step deep Q-learning except for algorithm steps n. These setups will be used for all of the simulation studies except when reporting the performance sensitivity with respect to certain parameters. In particular, the performance of different load forecasting steps K and the length of n-step return will be shown in the next subsection.

Table II: Hyperparameters of multi-step deep Q-learning

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Load forecast steps K	24	Discount factor γ	0.99
Algorithm steps n	8	Greedy maximum $\bar{\epsilon}$	1.0
Number of hidden units	150	Greedy minimum $\underline{\epsilon}$	0.01
I_{target}	60	Greedy attenuation length	3840
Learning rate α	0.0001	Number of episode	200

B. Optimality and Robustness of the Proposed Algorithm

In this subsection, we first report the performance of DQN and our proposed algorithm. We select the MIQP algorithm with Gurobi 9.1 [18] solver as a baseline algorithm for the 5-unit UC problem. We also run the training process 10 times independently using the same hyperparameters to show the robustness of the proposed algorithm. Finally, we show the process of determining the optimal value of the hyperparameters K and the number of steps n.

The average daily operation cost of the validation days is reported after every training episode in Figure 2. The initial status of the units on the first validation day are identical during the training process. Here we ran 5 independent experiments with different random seeds and calculate the mean value and standard deviations cross the runs. From Figure 2 we can see that the average daily cost of validation days calculated by multi-step deep Q-learning (n-step Q) declines quickly as the training process proceeds and stabilizes at a lower level than DQN after 50 training episodes.

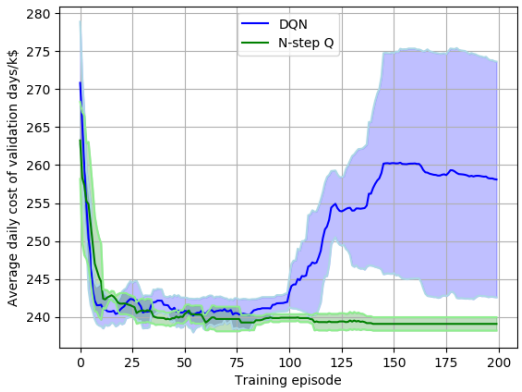

Figure 2: Average daily operation cost of validation days during the training process.

After training, we use the testing dataset to evaluate the algorithm performance. The parameters of the neural networks of DQN and n-step Q that minimize the average daily operation cost of validation days are used for testing. The daily operation cost of testing days calculated by DQN, multi-step

deep Q-learning, and MIQP are summarized in Table III. The percentage deviations of the DQN and the n-step Q-learning's performance from the MIQP, δ_1 and δ_2 are also reported. Due to the space limitation, only the daily costs of the first week and the total cost of 14 days are shown. The MIP gap of Gurobi solver is set to 0.0001%, so we can assume that the percent deviations in Table III are equal to the gap between the results of DQN or the proposed algorithm and the optimal value. As shown in Table III, the daily operation cost and percent deviation of the proposed algorithm is smaller than that of the DQN. The computing time of the 14 testing days of DQN, multi-step deep Q-learning, and MIQP are 1.61s, 1.37s, and 8.37s respectively. Note that the optimization periods of MIQP in table III are two days and we extract the operation cost of the first day from the optimization result.

Table III: Daily operation cost of DQN, n-step Q and MIQP

Day	DQN ($)	n-step Q ($)	MIQP ($)	$\delta_1(\%)$	$\delta_2(\%)$
1	238,714	238,653	238,153	0.24	0.21
2	215,132	212,440	209,788	2.55	1.26
3	208,069	205,525	203,356	2.32	1.07
4	228,218	227,506	226,968	0.55	0.24
5	228,584	227,704	227,072	0.67	0.28
6	227,857	225,512	225,082	1.23	0.19
7	227,477	225,297	224,952	1.12	0.15
14-Total	3,109,900	3,086,561	3,070,433	1.29	0.53

To verify the robustness of the proposed algorithm, we perform 10 independent experiments with different random seeds. The percent deviations of the daily costs of the 14 testing days between multi-step deep Q-learning and MIQP is shown is Figure 3. The daily costs of the testing days in 10 experiments are identical expect for two runs in most days.

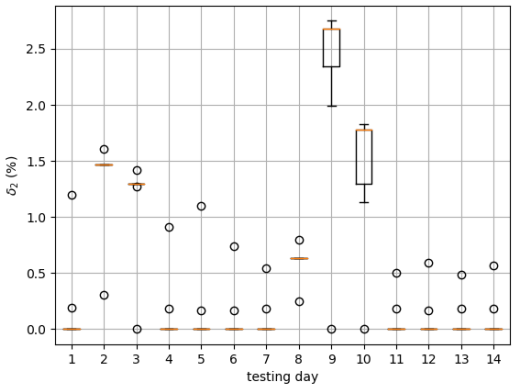

Figure 3: Box plot of testing days' operation costs.

Table IV: Total cost of the 14 testing days under different hyperparameters

($) n \ K	4	9	14	24
4	3,100,952	3,092,026	3,104,284	3,088,305
8	3,095,295	3,087,631	3,090,681	3,086,205
12	3,091,644	3,102,634	3,096,974	3,090,424
16	3,102,398	3,090,878	3,089,610	3,087,306
20	3,096,879	3,089,342	3,088,598	3,087,165
24	3,095,571	3,087,700	3,094,630	3,086,634

Table IV shows the total costs of the 14 testing days using different values for hyperparameters K and n. Also, we run 5 independent experiments with different seeds to calculate the average total operational cost. We can see that when $K = 24$ and $n = 8$, the proposed algorithm yields the lowest operational cost of $3,086,205.

V. Conclusion

This paper proposes a multi-step deep reinforcement learning algorithm to solve the UC problem, which is formulated as a Markov decision process. We use deep networks to parameterize an approximate action-value function and design an algorithm to determine the feasible action space. Numerical studies on a 5-unit UC test case show that our proposed algorithm outperforms the DQN significantly and almost matches the global optimal solutions identified by MIQP. The advantage of the proposed multi-step deep Q-learning algorithm over MIQP is in terms of computation time.

References

[1] T. Senjyu, K. Shimabukuro, K. Uezato, and T. Funabashi, "A fast technique for unit commitment problem by extended priority list," *IEEE Transactions on Power Systems*, vol. 18, no. 2, pp. 882–888, 2003.

[2] Z. Ouyang and S. Shahidehpour, "An intelligent dynamic programming for unit commitment application," *IEEE Transactions on power systems*, vol. 6, no. 3, pp. 1203–1209, 1991.

[3] M. Carrión and J. M. Arroyo, "A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem," *IEEE Transactions on power systems*, vol. 21, no. 3, pp. 1371–1378, 2006.

[4] M. P. Nowak and W. Römisch, "Stochastic Lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty," *Annals of Operations Research*, vol. 100, no. 1, pp. 251–272, 2000.

[5] G. Purushothama and L. Jenkins, "Simulated annealing with local search-a hybrid algorithm for unit commitment," *IEEE transactions on power systems*, vol. 18, no. 1, pp. 273–278, 2003.

[6] S. A. Kazarlis, A. Bakirtzis, and V. Petridis, "A genetic algorithm solution to the unit commitment problem," *IEEE transactions on power systems*, vol. 11, no. 1, pp. 83–92, 1996.

[7] W. Wang, N. Yu, Y. Gao, and J. Shi, "Safe off-policy deep reinforcement learning algorithm for volt-var control in power distribution systems," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3008–3018, 2020.

[8] G. Dalal and S. Mannor, "Reinforcement learning for the unit commitment problem," in *2015 IEEE Eindhoven PowerTech*. IEEE, 2015, pp. 1–6.

[9] N. K. Navin and R. Sharma, "A fuzzy reinforcement learning approach to thermal unit commitment problem," *Neural Computing and Applications*, vol. 31, no. 3, pp. 737–750, 2019.

[10] F. Li, J. Qin, and W. X. Zheng, "Distributed Q-learning-based online optimization algorithm for unit commitment and dispatch in smart grid," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 4146–4156, 2019.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.

[12] A. J. Wood, B. F. Wollenberg, and G. B. Sheblé, *Power generation, operation, and control*. John Wiley & Sons, 2013.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[14] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[15] C. Watkins, "Learning from delayed rewards," *PhD thesis, King's College, University of Cambridge*, 1989.

[16] Y. Gao, W. Wang, and N. Yu, "Consensus multi-agent reinforcement learning for volt-var control in power distribution networks," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3594–3604, 2021.

[17] N.-P. Yu, C.-C. Liu, and J. Price, "Evaluation of market rules using a multi-agent system method," *IEEE Transactions on Power Systems*, vol. 25, no. 1, pp. 470–479, 2009.

[18] Gurobi. (2021, May) The gurobi optimization website @ONLINE. [Online]. Available: https://www.gurobi.com/products/gurobi-optimizer/