frontiers in Energy Research

METHODS
published: 23 March 2022
doi: 10.3389/fenrg.2022.858895

Microgrid Energy Management Strategy Base on UCB-A3C Learning

Yanhong Yang[1], Haitao Li[2], Baochen Shen[2], Wei Pei[1] and Dajian Peng[1]*

[1]Institute of Electrical Engineering, Chinese Academy of Sciences, Beijing, China, [2]Faculty of Information Technology, Beijing University of Technology, Beijing, China

The uncertainty of renewable energy and demand response brings many challenges to the microgrid energy management. Driven by the recent advances and applications of deep reinforcement learning a microgrid energy management strategy, i.e., upper confidence bound based advantage actor-critic (A3C), is proposed to utilize a novel action exploration mechanism to learn the power output of wind power generation, the price of electricity trading and power load. The simulation results indicate that the UCB-A3C learning based energy management strategy is better than conventional PPO, actor critical and A3C algorithm.

Keywords: microgrid, energy management, A3C, UCB, edge computing

OPEN ACCESS

Edited by:
Junhui Li,
Northeast Electric Power University, China

Reviewed by:
Yassine Amirat,
ISEN Yncréa Ouest, France
Qifang Chen,
Beijing Jiaotong University, China

****Correspondence:***
Haitao Li
lihaitao@bjut.edu.cn

Specialty section:
This article was submitted to Smart Grids, a section of the journal Frontiers in Energy Research

Received: *20 January 2022*
Accepted: *03 March 2022*
Published: *23 March 2022*

Citation:
Yang Y, Li H, Shen B, Pei W and Peng D (2022) Microgrid Energy Management Strategy Base on UCB-A3C Learning. Front. Energy Res. 10:858895. doi: 10.3389/fenrg.2022.858895

INTRODUCTION

In the context of transition towards sustainable and cleaner energy production, microgrid (MG) has become an effective way for tackling energy crisis and environmental pollution issues. The microgrid is a small-scale energy system consisting of distributed energy sources and loads, which can operate independently from, or in parallel with, the main power grid (Yang et al., 2018). A typical microgrid system is illustrated in **Figure 1A**, which includes distributed generation resources (DERs), energy storage systems (ESS) and electric loads. Establishment of microgrid by integrating local renewable energy sources and loads, provides reliability guarantee for local service and strengthen grid resilience, and is a significant step towards Smart Grids (Chen et al., 2020; Lee, 2022; Li et al., 2020).

With the fluctuation of renewable energy supply and the uncertainty of power load change, how to manage microgrid more efficiently is a major challenge. When dealing with microgrid energy storage management, model-based algorithms such as particle swarm algorithm and ant colony algorithm have been proposed to solve this problem (Zhang et al., 2019). However, the dynamic characteristics of microgrid and the interaction between its components are described by building a model, which is not portable and scalable in practical application.

Recently, since the requirement of an explicit system model can be relaxed by learning-based scheme, this scheme has been introduced as an alternative to model-based approaches, and is used to improve the scalability of microgrid management (Kim et al., 2022; Fan et al., 2021; Nakabi and Toivanen, 2021; Pourmousavi et al., 2010; Yu et al., 2019). The deep reinforcement learning paradigm, which treats the microgrid as a black box, is the most promising learning-based method to find an optimal microgrid energy management strategy from interactions with it. Recently, microgrid energy management adopting a variety of DRL methods, has been investigated, such as DQN (S.A et al., 2010), SARSA (Ming et al., 2017), and Double DQN (Mnih et al., 2016).

Furthermore, literature (Finland, 2018) has explored A3C algorithm based on the policy gradient, and demonstrated that it has better performance than the value function based DRL algorithms in

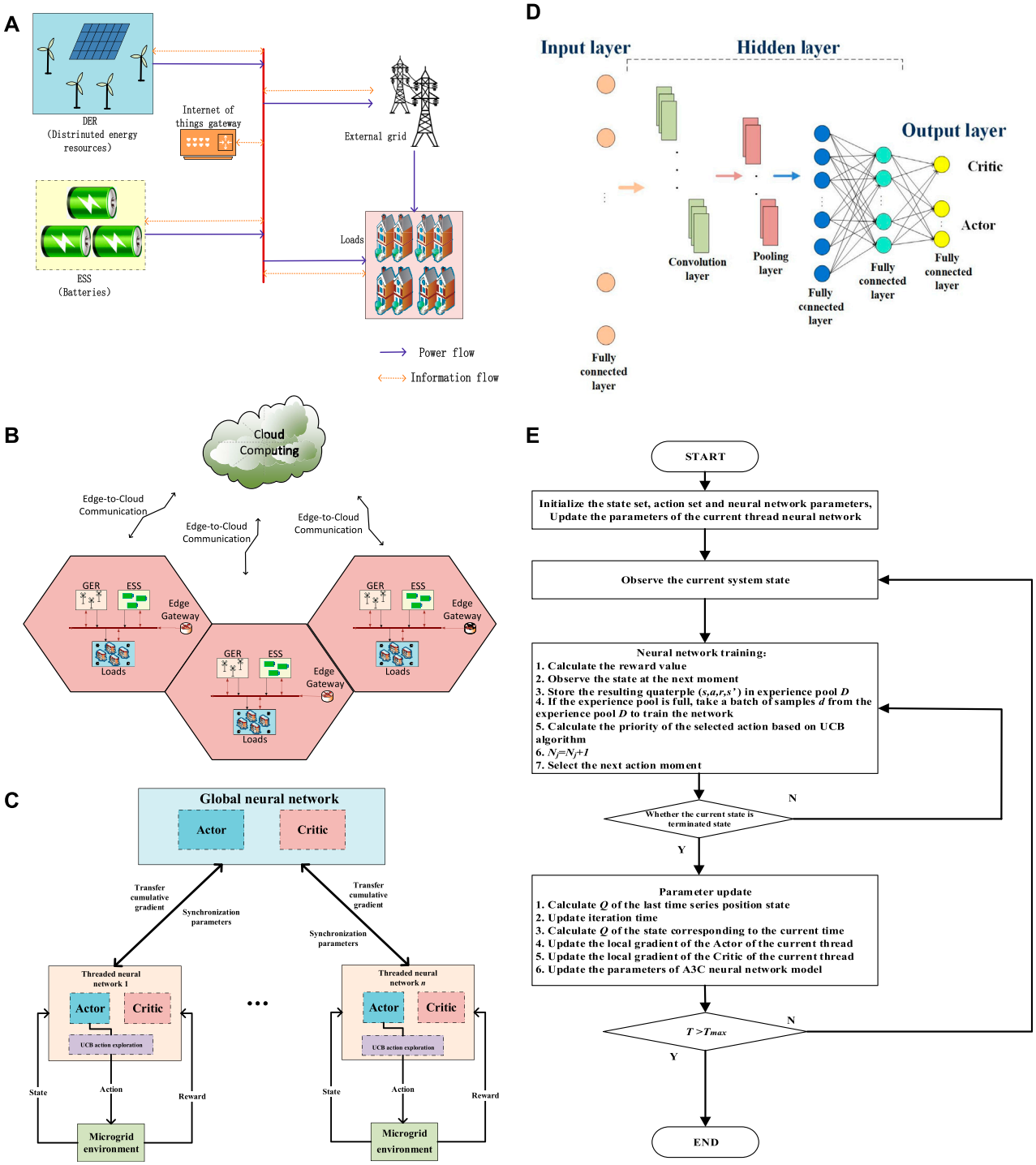

FIGURE 1 | (A) is the microgrid model, **(B)** is the microgrid edge computing architecture, **(C)** is the UCB-A3C algorithm, **(D)** is the neural network structure, **(E)** is the microgrid energy management strategy.

different microgrid operation scenarios. However, note that the conventional A3C approach adopts the heuristic ε-greedy method in the process of exploration. It always chooses the current best action with probability 1-ε or choose action randomly with probability ε. This greedy exploration approach leads to computation complexity proportional to time and the learning performance may be deteriorated (Dong et al., 2021). Based on these observations, an improved A3C learning algorithm with the novel exploration mechanism is proposed to deal with this problem in the learning process, which is benefit

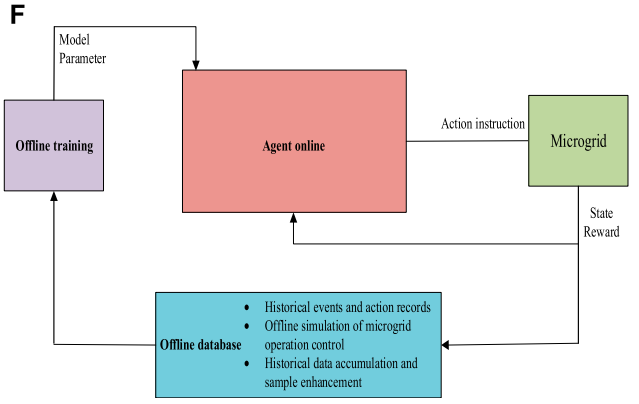

FIGURE 1 | Continued.

to real electricity price and renewable energy production in microgrid energy management.

On the other side, with the continuous advancement of new generation information and communication technologies in recent years, the newly emerging technologies, such as Internet-of-Things (iot), cloud computing and big data analytics, are deeply integrated to form the energy internet and enable new microgrid operational opportunities. Since the iot devices generate tremendous data during the microgrid operation, and they have the limited computation and storage capacity, it is necessary to introduce the cloud computing facilities to cope with these data. However, centralizing data processing in the cloud side would result in significant communication overhead and delay.

To address this issue, the edge computing architecture, which performs computational tasks at the edge of the communication network, is able to bring the cloud computing in close to the internet of thing devices (Finland, 2018). It provides the opportunity to integrate the training and inference process of microgrid energy management based on DRL at the edge, which is different from the conventional centralized cloud computing platform for mission-critical and delay-sensitive applications. The edge computing architecture and corresponding cloud-edge coordination mechanism enable the edge gateway to execute the decision-making tasks for timely energy management. Therefore, the edge computing is considered a promising solution to significantly reduce communication delay, improve microgrid energy management performance and bring distributed intelligence for the microgrid system. And then, with the help of the UCB-A3C learning algorithm, we present an intelligent energy management policy in this industrial edge computing environments.

The main contributions of this paper can be summarized as follows.

1) We integrate energy iot communication and cloud-edge coordination and present an edge computing architecture for microgrid energy management and optimization problem. Further, we designed a Markov decision process (MDP) with an objective of minimizing the daily operating cost to model this energy management issue.
2) To handle the formulated MDP optimization problem, we propose UCB based A3C learning algorithm with gross margin reward function, which can utilize a novel action exploration mechanism to learn the power output of wind power generation, the price of electricity trading and power load.

The rest of this paper is organized as follows. *System Model* describes the microgrid energy management architecture with edge computing and the MDP model of energy optimization problem. Then, the UCB based A3C learning algorithm with better learning efficiency is proposed in *UCB-A3C Based Energy Management*. Further on, an energy management approach based on the proposed UCB-A3C learning algorithm is proposed in this section. The performance evaluation of the UCB-A3C based energy management strategy is analyzed with simulations in *Performance Evaluation*. Finally, the conclusions are drawn in *Conclusion*.

SYSTEM MODEL

The proposed microgrid energy management architecture with edge computing is illustrated in **Figure 1B**. It integrates energy iot platforms with edge computing to implement ubiquitous sensing, computing and communication, and can effectively deploy learning based microgrid operation functionalities. This architecture consists of microgrid equipment layer, edge layer and cloud layer. The microgrid equipment layer is composed of various power components and is responsible for supplying the electricity to meet the local demand. We assume that the microgrid includes a group of TCLs, a wind-based DER, a communal ESS and a group of residential price-responsive loads, and these components are managed by edge gateway. Moreover, the microgrid uses these components to trade electricity with the main network to achieve a balance between supply and demand. In this process, if the power required by the power load component is greater than the power generation capacity of wind power generation, it adjusts the energy storage component to dynamically balance the power purchased from the main grid. If the power required by the power load component is less than the power generation capacity of wind power generation, it adjusts the power sold by the energy storage component to the main grid for dynamic balance.

The edge layer, which includes edge platforms, edge gateways and edge services, is located between the cloud platform and the underlying physical equipment layer. It is the key part of the entire architecture and provides functions such as storage, computing and application on the edge side. The hardware platform of edge layer is edge gateway, which is composed of communication modules, storage units and computing units, and is leveraging to perform data acquisition, transmission and microgrid equipment control. Edge gateway can support communication protocols such as

RS485, WiFi and 5G, as well as network transmission protocols such as HTTPS and MQTT. Under the co-scheduling of the edge platform, edge gateway can obtain microgrid operational states and control the microgrid equipment according to the instructions from the cloud server.

Edge computing platform is a software environment that is used to write and run software applications and is operated by the distributed edge gateways, and it is usually a standardized interoperability framework deployed on edge gateway to provide plug and play functions for iot sensors. All kinds of micro services for the microgrid based on some popular edge computing platform, such as EdgeX Foundry and KubeEdge, can run on edge platform. The edge platform aggregates microgrid equipment data which is collected from edge gateway. Meanwhile, it may store temporary sensor data, upload long-term data to the cloud center for data monitoring, analysis, storage and visualization, and receive the control instructions from the cloud side at the same time. In addition, edge service as the external interface of the whole microservice system, it collects iot equipment resources in the edge layer and provides services to users, accesses RESTful requests and forwards them to internal microservices. Moreover, in order to improve the capacity of computing and storage, edge service takes advantage of cloud-edge collaboration to cooperate edge resources with cloud resources, and supports powerful expansion capability to implement service mapping, request parsing, encryption and decryption, and authentication.

The cloud layer uses the cloud platform to provide various cloud services, we can deploy different cloud infrastructure environments, such as public cloud, private cloud or hybrid cloud, for different scale microgrid using the elastic expansion capability of the cloud platform, and run deep reinforce learning based intelligent microgrid energy management strategy in the cloud side. Due to the cloud platform can provide sufficient computation and storage resources, the exhaustive analysis of the massive historical data through a model training process is supported by cloud service. The well-trained model can be further transfer to the edge computing layer to implement the local microgrid energy management functionalities.

Specifically, based on the physical architecture of microgrid energy management given above, next we describe the theoretical model of energy management. Note that the agent selects an action under the state and the environment gets the next state in the microgrid scenario, and that the next state of the agent only depends on the current state and the action, and it is not related to previous states and actions. Consequently, microgrid energy management can be formulated as MDP problem. In general, the state space S, action space A, and reward function R are used for the agent-environment interaction modeling of the MDP. We define that the state space includes exogenous state component and controllable state component, and the action space includes the energy deficiency action, price action, and TCL action. After the agent transfers from state S_t to state s_{t+1}, it received the immediate reward R_t when an action α is given. With an objective of minimizing the daily operating cost, the reward function, as gross margin from operations, is given by:

$$\boldsymbol{R_t = Rev_t - Cos_t} \tag{1}$$

where $Rev_t = P_{load}\sum_{i=0}^{N_{loads}} L_{load}^{i,t} + P_{tcls}\sum_{i=0}^{N_{TCLs}} L_{tcl}^{i,t} + (P_{down}^t - P_{sold}^t)E_{sold}^t$ is the microgrid revenues from selling electricity to the external grid, $Cos_t = P_{cost}G_t + (P_{up}^t + P_{puch}^t)E_t^P$ is the costs related to purchases from the power generation and external grid. P_{loads} is the price of the price-responsive loads, N_{load} is the number of the price-responsive load. $L_{load}^{i,t} = L_{b,t} - SL_t^i + PB_t^i$ represents the power consumption of the price-responsive loads at time t. P_{tcls} is the price of the direct controllable loads, N_{TCLs} is the number of the direct controllable loads. $L_{tcl}^{i,t}$ represents the power consumption of the direct controllable loads at time t, it can be calculated by $L_{tcl}^{i,t} = P_{tcl}u_{control}^{i,t}$. P_{sold}^t and P_{purch}^t are the power transmission costs respectively for exporting to and importing from the external grid. P_{up}^t is the up-regulation price and P_{down}^t is down-regulation price. The power generation cost is P_{cost}. The energies purchased, sold to, and generated from the external grid are E_t^P, E_{sold}^t and G_t respectively.

UCB-A3C BASED ENERGY MANAGEMENT

In view of the fact that the dimension of state space and action space in microgrid are large. To solve this MDP problem, the A3C learning algorithm, which is a state-of-the-art actor-critic method that exploits multi-threading to create several learning agents, is effective to handle the large scale decisions-making problem and is considered in this paper.

A3C Algorithm

Different from the classical actor-critic algorithm with only one learning agent, the A3C method adopts asynchronously parallel learning of multiple actor on different threads. The key advantage of this parallel learn scheme in different threads is that it breaks the interdependence of gradient updates and decorrelates past experiences gained by each learning agent, and it is an online learning algorithm and converges rapidly (Jia et al., 2015; Liu et al., 2019; Lee et al., 2020).

In A3C algorithm with a multi-threaded training framework, it has one global network consisting of actor network and critical network. These two neural networks have different function. To be specific, the policy gradient schemes is utilized by actor network to choose the action, and the parameterized policy with a set of actor parameters θ_a is defined by $\pi(a|s;\theta_a) = P(a|s,\theta_a)$, and the gradient-descent method is applied to update the parameters. The critic network evaluates each action from the actor network and learns the value function while multiple actors are trained in parallel. And in order to qualify the expected reward, the critic network estimates the state-value function $V(s_i;\theta_c)$ on account of state s with critic parameters θ_c.

During algorithm execution, each agent makes use of the value function to evaluate its policy to achieve the long-term cumulative reward. For the given policy π, the state-action value function, called Q-function, of state action pair (s, a) can be achieved by action a, it is defined as the expected reward by an action a in the state s,

TABLE 1 | The details of UCB-A3C algorithm is described.

Algorithm improvement A3C

Input: state values of each component of the microgrid
Output: action of each component of the microgrid
Initialization: discount factor μ, parameters of global A3C neural network θ, ω, parameters of current thread neural network $\theta^{'}, \omega^{'}$, the number of samples selected for training is d, the number of iteration rounds globally shared T, the maximum number of iteration rounds globally shared is T_{max}, initial time t_{start}
1: Reset the gradient update amount of public neural network, reset $d\theta = 0$, $d\omega = 0$
2: Update the parameters of the current thread neural network $\theta = \theta^{'}, \omega = \omega^{'}$
3: Observe the current system state s_t
4: Select action a_t base on strategy $\pi(a_t|s_t, \theta)$
5: Calculate the reward value r_t at the current time t and observe the state s_{t+1} at the next time
6: Store the resulting quaterple (s,a,r,s') in experience pool D
7: If the experience pool is full, take a batch of samples d from the experience pool D to train the network
8: Calculate the priority of the selected action $p = acts\_prob + \tau\sqrt{\frac{\ln(\varepsilon+\sigma)}{N_j}}$
9: $N_j = N_j+1$
10: Select the next action moment $a_{t+1} = argmax\, p$
11: $t \leftarrow t+1, T \leftarrow T+1$
12: Determine whether the current state of s_t is a terminated state, if not, return to step 5
13: Calculate $Q(s_t, t)$ of the last time series position state s_t
14: *For* $i \in (t-1, t-2, t-3, \ldots t_{start})$
15: Calculate $Q(s_i, i)$ of the state s_i corresponding to the current time t
16: Update the local gradient $\theta^{'}$ of the current thread
17: Update the local gradient $\omega^{'}$ of the current thread
18: end for
19: Update the neural network parameters (θ, ω)
20: Until $T > T_{max}$, otherwise, return to step 3

$$\boldsymbol{Q_\pi(s,a)} = \mathbb{E}[(\boldsymbol{R_t}|\boldsymbol{s_t} = \boldsymbol{s}, \boldsymbol{a_t} = \boldsymbol{a})] \tag{2}$$

and the state value function

$$\boldsymbol{V_\pi(s)} = \mathbb{E}[(\boldsymbol{R_t}|\boldsymbol{s_t} = \boldsymbol{s})] \tag{3}$$

where the expectation E (·) is taken over all possible the state-action transitions following the policy π. For the policy and the value function A3C learning algorithm, the parameters are updated by the n-step reward and the reward is defined as:

$$\boldsymbol{r_t} = \sum_{i=0}^{n-1} \boldsymbol{\gamma_i r(s_{t+i}, a_{t+i})} + \boldsymbol{\gamma_n V(s_{t+n}, \theta_c)} \tag{4}$$

where γ is the discount factor.

For update rule of A3C, it is desire to the agent not only learns how good the action is, but also learn how much better than expected. And the policy gradient scheme is adopted to perform parameters update for the A3C algorithm. However, high variance may be introduced by the policy gradient in the critic network. To deal with the problem, the $Q(s, a)$ function in the policy gradient process has been replaced by the advantage function As, t, and As, t is given by:

$$\begin{aligned}\boldsymbol{A(s,t)} &= \boldsymbol{r(t)} - \boldsymbol{V(s_t)} \\ &= \boldsymbol{R_t} + \boldsymbol{\gamma R_{t+1}} + \ldots + \boldsymbol{\gamma_{n-1} R_{t+n-1}} + \boldsymbol{\gamma_n V(s_{t+n})} - \boldsymbol{V(s_t)}\end{aligned} \tag{5}$$

where $V(s_{t+n})$ and $V(s_t)$ are the sate value function in the state s_{t+n} and s, respectively.

In the A3C learning framework, there are two loss functions and they are associated with the outputs of deep neural network, and all the actor-learners update the state value function $V(s)$ and the policy $\pi(s, a)$ by the gradient loss. The actor loss function is defined as:

$$\boldsymbol{L(\theta_a)} = \boldsymbol{log\pi(a_t|s_t, \theta_a)(r_t - V(s_t, \theta_c))} + \boldsymbol{\theta G(\pi(s_t, \theta_a))} \tag{6}$$

where θ is the hyperparameter and $G(\pi(s_t, \theta_a))$ is the entropy which is used to encourage exploration and discourage premature convergence to a suboptimal policy. The accumulated gradient of the $L(\theta_a)$ is expressed as:

$$\boldsymbol{d\theta_a} = \boldsymbol{d\theta_a} + \nabla_{\theta'_a} log\, \pi(\boldsymbol{a_t}|\boldsymbol{s_t}, \boldsymbol{\theta'_a})\boldsymbol{A(s,t)} + \boldsymbol{\theta}\nabla_{\theta'_a}\boldsymbol{G}(\boldsymbol{\pi}(\boldsymbol{s_t}, \boldsymbol{\theta'_a})) \tag{7}$$

where $\theta_a^{'}$ is the thread-specific parameter in actor network.

Similarly, the loss function in critic network is given by:

$$\boldsymbol{L(\theta_c)} = (\boldsymbol{r_t} - \boldsymbol{V(s_t, \theta_c)})^2 \tag{8}$$

and the accumulated gradient of $L(\theta_c)$ is defined as:

$$\boldsymbol{d\theta_c} = \boldsymbol{d\theta_c} + \nabla_{\theta_c^{'}}(\boldsymbol{r_t} - \boldsymbol{V(s_t, \theta_c)})^2 \tag{9}$$

where $\theta_c^{'}$ is the thread-specific parameter in critic network. In order to achieve the loss function minimization in our presented A3C framework, the standard noncentered RMSProp algorithm (Tijmen and Geoffrey, 2012) is utilized to perform training until the accumulated gradients shown in **Eqs 7**, **9** is updated.

Consider that the sufficient exploration is needed to avoid a suboptimal policy with worse reward and the exploitation adopts the policy with the best reward, the optimal learning strategy, which can implement the balance between exploration and exploitation, is expected to be achieved.

Proposed UCB-A3C Algorithm

To further improve the performance of the A3C algorithm, this paper is leveraging the idea of the UCB algorithm. Firstly, the agent selects an action through UCB exploration and executes it.

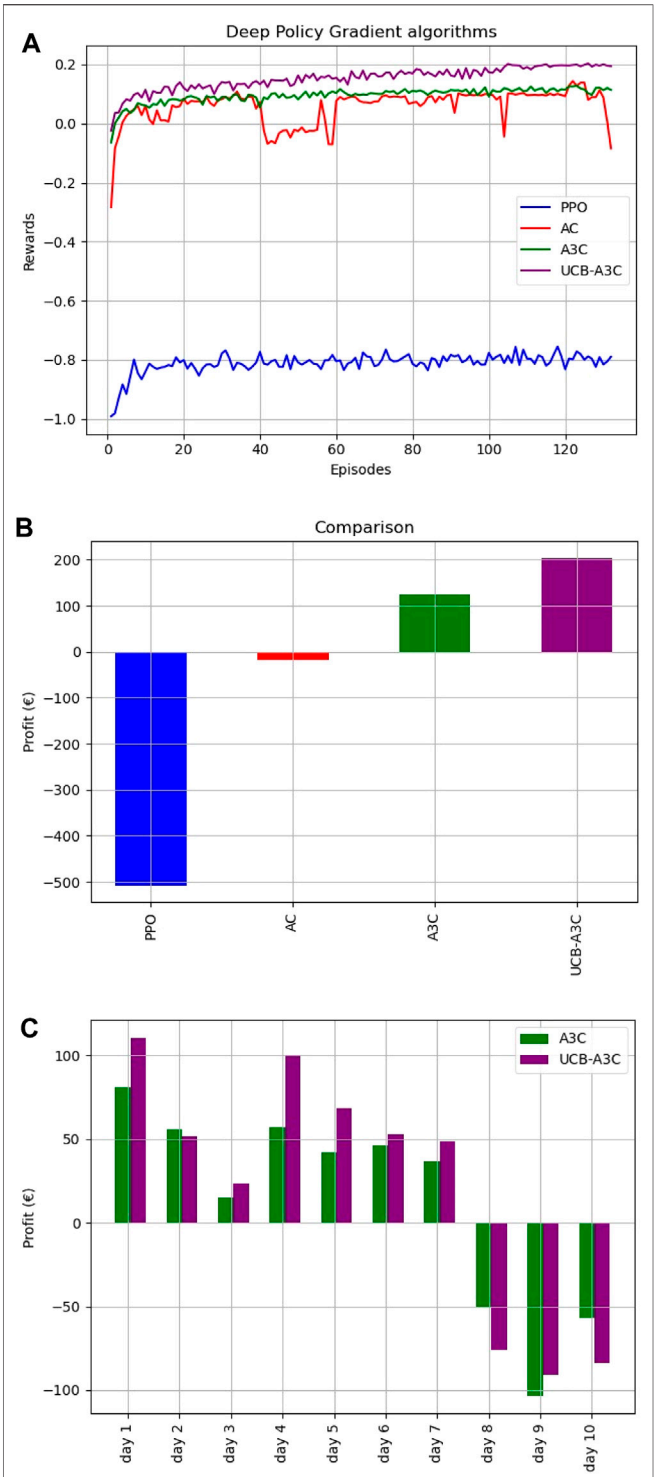

FIGURE 2 | Analysis of the UCB-A3C algorithm, **(A)** is the reward of the UCB-A3C algorithm, **(B)** is the total economic profits, **(C)** is the daily economic profits.

Then, after sampling from the experience pool and calculating the loss function, the priority of action is calculated. Finally, the priority will be assigned to the action to be performed. The priority of action is calculated by the following equation:

$$p = acts\_prob + \tau\sqrt{\frac{\ln(\varepsilon + \sigma)}{N_j}} \tag{10}$$

where N_j represents the number of the jth action is selected, $acts\_prob$ is the probability value returned by the actor network output, τ is the parameter that adjusts the influence of priority on its selection action, ε is the parameter that keeps decreasing, and σ is the parameter for ε correction. The second term in **Eq. 10** is the confidence factor. In the initial stage of the algorithm, the confidence factor is large, so it has a great impact on the priority. With the progress of training, the time step t continues to increase, and the influence of confidence factor will gradually decrease, so as to increase the chance of relative attempt and ensure the diversity of samples. At time t, if an action has been selected many times, the higher the reward value of the action, the greater the probability of being continued. If an action is selected few times, its confidence factor will be higher and the probability of being continued will be lower. When the algorithm reaches the convergence state, the benefit of optimal action selection can be maximized. As mentioned above, the framework of the UCB-A3C algorithm is shown in **Figure 1C**.

Besides, we also carefully design the neural network (NN) structure of the UCB-A3C algorithm. The input layer of NN is composed of 107 neurons, corresponding to the input 107 environmental states. The hidden layer is designed as a combination of convolutional layer, pooled layer and fully connected layer. After the data is input through the input layer, the data is convolved through a convolution layer. The convolution layer adopts a 3×3 convolution kernel. After output data from the convolution layer, the global average pooling layer is used for data pooling. Then the data is output to actor and critic network through the fully connected layer of two layers with the number of neurons being 200 and 100, respectively. The actor network is designed as a fully connected layer with the number of neurons being 80, while the critic network is designed as a fully connected layer with the number of neurons being 1. The structure of the neural network is shown in **Figure 1D**.

Consequently, the details of our proposed UCB-A3C algorithm is described in **Table 1**, and the flowchart of the proposed algorithm is shown in **Figure 1E**.

Clearly, the proposed algorithm indicates that the probability of the selected action with larger reward is effectively increased with the help of UCB.

TABLE 2 | The simulation parameters.

Parameter	Numerical
Maximum capacity of ESS	500 KWh
Charging power of ESS	250 KW
Discharge power of ESS	250 KW
Generation cost of DER	32€/MW
Generation capacity of DER	data source [Oy, 2018]
Number of directly controllable loads	100
The quantity of non-directly controllable loads	150
Electricity markets cut prices	data source [Fingrid Open Datasets., 2018]
Electricity markets raised prices	data source [Fingrid Open Datasets., 2018]

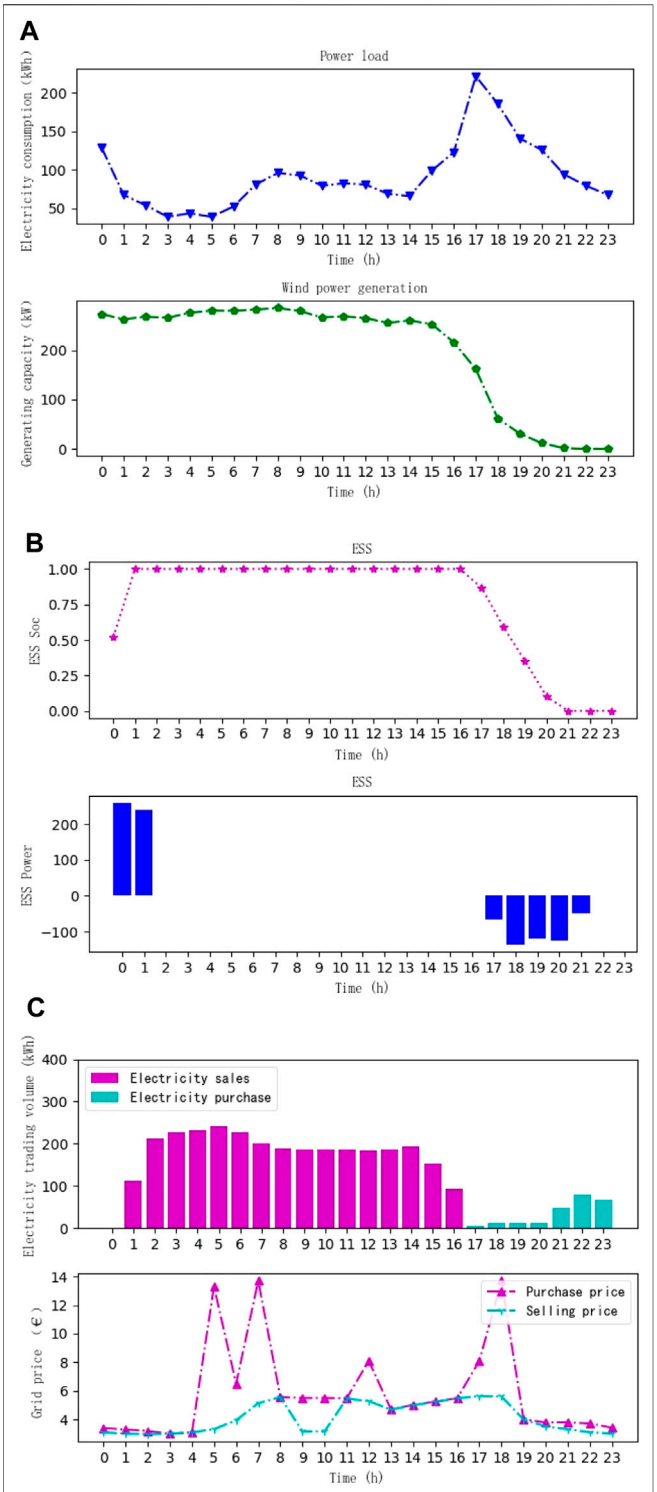

FIGURE 3 | The UCB-A3C algorithm is utilized to implement the microgrid energy management, **(A)** is the electric loads and distributed energy resource, **(B)** is the energy storage system, **(C)** is the electricity transaction volume and price change.

Implementation of Microgrid Energy Management

UCB-A3C based microgrid energy management consists of offline and online stages. The offline stage stores the action records of microgrid operation and historical events, improves and perfects the data set through historical data accumulation, performs the offline simulation of microgrid operation control to train the agent, and updates the agent model and parameters for the use of online agents. When the microgrid works in real time in the online stage, the agent calculates the output action and control command according to the state variables and rewards fed back by the microgrid. Moreover, in line with the control command from cloud server, the microgrid operates and feeds back the updated status and reward to the online agent, and stores it in the edge computing platform. The implementation of microgrid energy management strategy based on UCB-A3C is shown in **Figure 1F**.

PERFORMANCE EVALUATION

In order to verify the proposed microgrid energy management strategy based on UCB-A3C, a simulation model, as shown in **Figure 2**, is a built-in Python environment, and the simulation parameters are shown in **Table 2**.

In our simulation, PPO, Actor Critic, A3C and UCB-A3C algorithm are respectively used for training, and the obtained reward value is shown in **Figure 2A**. It can be observed that the UCB-A3C algorithm has higher reward in the learning process than other algorithms.

And that, the PPO, Actor-Critic, A3C and improved A3C algorithm were respectively used to carry out the economic profits of microgrid energy control, and the achieved total 10 days' economic profits is shown in **Figure 2B**. It can be found that the economic profits obtained by the UCB-A3C algorithm based microgrid energy management are greater than those obtained by the other three algorithms.

Simultaneously, the daily economic profits of the A3C algorithm and the UCB-A3C algorithm for ten consecutive days are compared, as shown in **Figure 2C**. It can be seen from the figure the UCB-A3C algorithm is superior to the A3C algorithm in six of the 10 days of revenue, which has better energy management efficiency.

Further, we make use of UCB-A3C algorithm to optimize microgrid energy management, and the predicted data of power generation and consumption of wind power generation components and power load components are shown in **Figure 3A**. At this time, the energy storage system is charged from the 0^{th} to 1st hours and discharged from the 17th to 21st hours in **Figure 3B**. In the energy trading market, electricity is mainly sold, and the trading price changes with the trading electricity is shown in **Figure 3C**.

To sum up, the improved A3C algorithm can carry out efficient energy coordination management on the microgrid, and then efficiently trade electricity with the power grid, so as to achieve the purpose of reasonable distribution of electricity, improve economic profits, and reduce the power loss in the process of power distribution.

CONCLUSION

Microgrids are an effective way to deal with flexible access to renewable energy and varying power loads. In order to deal with such volatility and uncertainty, this paper proposes an A3C algorithm based on UCB exploration mechanism. According to the simulation, we have validated that the proposed UCB-A3C algorithm can adapt the constantly changing microgrid environment, learn the efficient energy management strategy, and provide a more economical scheme for the microgrid operation, so as to achieve the purpose of reducing the economic cost. Moreover, UCB-A3C learning algorithm based microgrid energy management has solved the unscalable application and repeated development problems of traditional domain experts. However, in practical application, there are still some areas that need to be further improved due to its long training time and great dependence on training data in the learning process. Therefore, it is the focus of future research to solve the above problems in order to better apply deep reinforcement learning to microgrid energy management.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

YY is responsible for algorithm research. HL is responsible for algorithm research. BS is responsible for simulation. WP is responsible for algorithm design. DP is responsible for writing paper.

FUNDING

This work is supported by the National Natural Science Foundation of China (No. U2066211) and the Youth Innovation Promotion Association, Chinese Academy of Sciences (No. 2021136).

REFERENCES

Chen, H., Guan, L., Lu, C., et al. (2020). Multi-objective Optimal Dispatch Model and its Algorithm in Isolated Microgrid with Renewable Energy Generation as Main Power Supply. *Power Syst. Technol.*.

Dong, W., Yang, Q., Li, W., and Zomaya, A. Y. (2021). Machine-learning-based Real-Time Economic Dispatch in Islanding Microgrids in a Cloud-Edge Computing Environment. *IEEE Internet Things J.* 8, 13703–13711. doi:10.1109/JIOT.2021.3067951

Fan, L., Zhang, J., He, Y., Liu, Y., Hu, T., and Zhang, H. (2021). Optimal Scheduling of Microgrid Based on Deep Deterministic Policy Gradient and Transfer Learning. *Energies* 14, 584. doi:10.3390/en14030584

Fingrid Open Datasets (2018) Fingrid Open Datasets. Available: https://data.fingrid.fi/open-dataforms.

Oy, Fortum (2018). *Wind Farm Data*. Finland.

Jia, H., Wang, D., Xu, X., et al. (2015). Research on Some Key Problems Related to Integrated Energy Systems. *Automation Electric Power Syst.*doi:10.7500/AEPS20141009011

Kim, J., Oh, H., and Choi, J. K. (2022). Learning Based Cost Optimal Energy Management Model for Campus Microgrid Systems. *Appl. Energ.* 311, 118630. doi:10.1016/j.apenergy.2022.118630

Lee, J., Wang, W., and Niyato, D. (2020). Demand-Side Scheduling Based on Multi-Agent Deep Actor-Critic Learning for Smart Grids. *Early Access*. doi:10.1109/SmartGridComm47815.2020.9302935

Lee, S., and Choi, D.-H. (2022). Federated Reinforcement Learning for Energy Management of Multiple Smart Homes with Distributed Energy Resources. *IEEE Trans. Ind. Inf.* 18, 488–497. doi:10.1109/TII.2020.3035451

Li, H., Wan, Z., and He, H. (2020). Real-time Residential Demand Response. *IEEE Trans. Smart Grid* 11, 4144–4154. doi:10.1109/TSG.2020.2978061

Liu, Y., Yang, C., Jiang, L., Xie, S., and Zhang, Y. (2019). Intelligent Edge Computing for IoT-Based Energy Management in Smart Cities. *IEEE Netw.* 33, 111–117. doi:10.1109/MNET.2019.1800254

Ming, M., Rui, W., Zha, Y., and Tao, Z. (2017). "Multi-Objective Optimization of Hybrid Renewable Energy System with Load Forecasting," in 2017 IEEE International Conference on Energy Internet (ICEI), Beijing, China, 17-21 April 2017. doi:10.1109/icei.2017.27

Mnih, Volodymyr., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). "Asynchronous Methods for Deep Reinforcement Learning," in International Conference on Machine Learning, ICML 2016.

Nakabi, T. A., and Toivanen, P. (2021). Deep Reinforcement Learning for Energy Management in a Microgrid with Flexible Demand. *Sustainable Energ. Grids Networks* 25, 100413. doi:10.1016/j.segan.2020.100413

Pourmousavi, S. A., Nehrir, M. H., Colson, C. M., and Wang, C. (2010). Real-Time Energy Management of a Stand-Alone Hybrid Wind-Microturbine Energy System Using Particle Swarm Optimization. *IEEE Trans. Sustain. Energ.* 1, 193–201. doi:10.13335/j.1000-3673.pst.2019.0298

Tijmen, T., and Geoffrey, H. (2012). Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude. *Coursera: Neural Networks Machine Learn.* 4.

Yang, Y., Pei, W., Huo, Q., Sun, J., and Xu, F. (2018). Coordinated Planning Method of Multiple Micro-grids and Distribution Network with Flexible Interconnection. *Appl. Energ.* 228, 2361–2374. doi:10.1016/j.apenergy.2018.07.047

Yu, L., Xie, W., Xie, D., Zou, Y., Zhang, D., Sun, Z., et al. (2020). Deep Reinforcement Learning for Smart home Energy Management. *IEEE Internet Things J.* 7, 2751–2762. doi:10.1109/JIOT.2019.2957289

Zhang, Z., Qiu, C., Zhang, D., et al. (2019). A Coordinated Control Method for Hybrid Energy Storage System in Microgrid Based on Deep Reinforcement Learning. *Power Syst. Technol.*

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.