Peer-to-Peer Networking and Applications (2024) 17:1702–1719
https://doi.org/10.1007/s12083-024-01663-5

Deep learning-based power usage effectiveness optimization for IoT-enabled data center

Yu Sun[1] · Yanyi Wang[1] · Gaoxiang Jiang[1] · Bo Cheng[1] · Haibo Zhou[1]

Received: 30 October 2023 / Accepted: 11 February 2024 / Published online: 22 March 2024

Abstract
The proliferation of data centers is driving increased energy consumption, leading to environmentally unacceptable carbon emissions. As the use of Internet-of-Things (IoT) techniques for extensive data collection in data centers continues to grow, deep learning-based solutions have emerged as attractive alternatives to suboptimal traditional methods. However, existing approaches suffer from unsatisfactory performance, unrealistic assumptions, and an inability to address practical data center optimization. In this paper, we focus on power usage effectiveness (PUE) optimization in IoT-enabled data centers using deep learning algorithms. We first develop a deep learning-based PUE optimization framework tailored to IoT-enabled data centers. We then formulate the general PUE optimization problem, simplifying and specifying it for the minimization of long-term energy consumption in chiller cooling systems. Additionally, we introduce a transformer-based prediction network designed for energy consumption forecasting. Subsequently, we transform this formulation into a Markov decision process (MDP) and present the branching double dueling deep Q-network. This approach effectively tackles the challenges posed by enormous action spaces within MDP by branching actions into sub-actions. Extensive experiments conducted on real-world datasets demonstrate the exceptional performance of our algorithms, excelling in prediction precision, optimization convergence, and optimality while effectively managing a substantial number of actions on the order of 10^{13}.

Keywords Data center · IoT · Power usage effectiveness · Deep learning · Large-sacle optimization

1 Introduction

Data centers significantly contribute to both global energy consumption and carbon emissions. As per estimations in [1], the worldwide energy consumption by data centers is projected to surge from 800 terawatt hours (TWh) in 2020 to a staggering 2967 TWh by 2030. The recent remarkable success of ChatGPT has ignited the proliferation of various generative AI models, thereby generating an unprecedented demand for computational capabilities. This surge in demand is expected to further propel the growth of data centers. Governments and data center operators are taking proactive measures to establish more energy-efficient data centers. For instance, Google has committed to achieving carbon neutrality by 2030, and Microsoft plans to attain carbon negativity by the same year [2]. Additionally, the Chinese government has mandated that newly constructed data centers must adhere to a power usage effectiveness (PUE) threshold of no more than 1.3 by the end of 2023, as outlined in the Three Year Plan [3]. In large data centers, chiller cooling systems are utilized to deliver cooling to server rooms, constituting a significant share (46%) of energy consumption within data centers [4]. Consequently, the optimization of chiller cooling systems has arisen as a promising approach to mitigate energy consumption, garnering noteworthy consideration in recent years.

This article is part of the Topical Collection: *Special Issue on Affordable and Clean Energy*
Guest Editors: Dajiang Chen, Ning Zhang, and Chunpeng Ge

✉ Haibo Zhou
haibozhou@nju.edu.cn

Yu Sun
yusun@smail.nju.edu.cn

Yanyi Wang
wyynju@foxmail.com

Gaoxiang Jiang
jgxwww@sina.cn

Bo Cheng
bocheng@smail.nju.edu.cn

[1] School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China

Current research typically adopts a two-step approach to optimize cooling systems. Initially, a theoretical system model is constructed to describe the physical system, following which various algorithms are applied to optimize energy consumption based on this system model. Traditional methods incorporating expert knowledge and thermodynamic principles fall short in capturing the intricacies of these systems, leading to suboptimal optimization results. In recent years, thanks to the rapid advancement of the Internet of Things (IoT), data centers have gained the ability to employ diverse sensors for extensive data collection from the physical environment. This collected data offers a precise real-time depiction of the data center's status, a concept verified in recent works [5, 6]. This lays the foundation for precise data-driven modeling in data centers. Deep learning, as a representative method in the data-driven domain, serves as a powerful tool for modeling complex relationships [7–10]. It offers an appealing alternative for achieving more accurate modeling and improved optimization results in the context of data center PUE optimization. Deep learning-based algorithms have found applications in modeling data centers [11–13], while for optimization purposes, deep reinforcement learning (DRL) based algorithms have also garnered significant attention [14–16]. Nevertheless, these prediction methods encounter challenges in effectively modeling complex data centers and managing long-term temporal dependencies. Hence, the need for more accurate prediction algorithms is evident, ultimately leading to superior optimization outcomes. In the context of actual data center operations, a multitude of parameters require control and optimization, resulting in a enormous optimization space. However, the existing DRL algorithms applied in data centers struggle to address such vast action spaces.

In this paper, we investigate the optimization of PUE in IoT-enabled data centers by employing deep learning-based algorithms. Initially, we develop a deep learning-based PUE optimization framework tailored to IoT-enabled data centers. We proceed to formulate the general PUE optimization problem, which is subsequently specified and simplified to focus on minimizing the energy consumption of chiller cooling systems. Recognizing the significance of precise modeling for optimization, we introduce a transformer-based prediction network designed to accurately forecast the energy consumption of chiller cooling systems. Building on this network, we further reconfigure the optimization problem into a Markov decision process (MDP). To address the enormous action space inherent in MDP, we propose a method to compress this space by branching actions into sub-actions, a technique we refer to as branching double dueling deep Q-network. Finally, we substantiate the exceptional performance of our proposed algorithms via comprehensive experiments conducted on real-world datasets. The primary contributions of this paper can be summarized as follows:

- We establish a deep learning-based framework for optimizing PUE in IoT-enabled data centers. We formulate the general PUE optimization problem, which is then refined and simplified to focus on minimizing the energy consumption of chiller cooling systems.
- We introduce a transformer-based prediction network (TPN) designed to forecast energy consumption in chiller cooling systems. This network excels in accurately capturing temporal dependencies, thereby establishing a robust foundation for subsequent optimization.
- We convert the original problem into an MDP and subsequently introduce the branching double dueling deep Q-network (BD3QN) to address the considerable action space within our problem. This approach branches the action space into sub-actions, handling substantial actions while retaining benefits in terms of both convergence and optimality.

The remainder of this paper is structured as follows. In Section 2, we survey state-of-the-art works concerning PUE optimization. In Section 3, we present the system model and articulate the formulated optimization problem. Section 4 elucidates the TPN designed for forecasting chiller cooling system energy consumption. In Section 5, we provide a detailed explanation of the proposed energy consumption optimization algorithm, BD3QN. Section 6 showcases the extensive results and insights obtained. Lastly, Section 7 offers the concluding remarks for this paper.

2 Related works

Early investigations into cooling systems [17–20] rely on thermodynamic principles, mechanical principles, empirical formulas, and expert knowledge. For instance, Ma et al. [17] devised an optimal strategy through a systematic approach that hinges on model-based performance prediction using various models to encompass comprehensive system components. They constructed the chiller model based on thermodynamic principles and heat/mass transfer processes, while the air handling unit model was formulated on empirical foundations. Sun et al. [18] utilized a complete simulation-based sequential quadratic programming approach to attain optimal control. This method enables the use of precise component models, as opposed to empirical ones. Lu et al. [19] optimized overall system performance by employing component characteristic analysis, mathematical models, and solving the problem through a modified genetic algorithm. Furthermore, Fang et al. [20] conducted an investigation into the simultaneous optimization of computing and cooling within data centers using a two-time-scale control method. This approach involves the optimization of cooling supplements through model predictive control. However, these traditional methods

fall short in capturing the complexity of intricate cooling systems, resulting in suboptimal, unstable, and insecure optimization outcomes [21].

Given the significant capability of deep learning-based methods to model intricate and nonlinear systems, their application in modeling and prediction within data centers has garnered substantial attention. Vu et al. [11] conducted an investigation into data-driven chiller plant energy optimization by incorporating domain knowledge into data analysis. They employed distinct deep-learning algorithms tailored to specific module types within a module-wise modeling approach. In the study by Yang et al. [12], the authors achieved precise PUE prediction through meticulous feature selection and analysis. By assessing the importance and sensitivity of these features, they optimized the condenser water flow of chillers, resulting in a reduction of PUE and energy savings verified through field testing in data centers. Considering the significance of temperature and humidity, the modeling and prediction of temporal correlations becomes essential. Consequently, Zhao et al. [13] employed a gated recurrent unit (GRU)-based approach relying on historical data to predict PUE.

Following the pioneering success of deep learning in the field of reinforcement learning [22], DRL has gained substantial recognition among researchers [23, 24]. Its adoption has also extended to various domains [25–27]. Within the context of optimizing energy consumption in data centers, notable advancements have recently emerged. Heimerson et al. [14] initiated their study by developing a data center model through computational fluid dynamics (CFD) and subsequently trained a DRL model based on this foundation. They specifically employed the proximal policy optimization algorithm, optimizing four flow and temperature setpoints of the computer room air handler. Similarly, Li et al. [15] harnessed DRL with the deep deterministic policy gradient algorithm to optimize five temperature setpoints within chiller cooling systems. Furthermore, actor-critic enabled DRL, as illustrated by Naug et al. [16], successfully optimized the temperature setpoint and air handling unit preheat/reheat discharge.

While some prior works have employed deep learning-based algorithms to predict and optimize data center energy consumption, new solutions are warranted for the following reasons. Firstly, existing prediction algorithms struggle to effectively handle the complexity and long-term temporal dependencies inherent in modeling data centers. This limitation results in unsatisfactory prediction performance, leading to suboptimal optimization outcomes. Secondly, existing DRL algorithms are primarily designed with the assumption of continuous and low-dimensional action spaces. In contrast, real-world data centers feature discrete and vast action spaces, rendering the application of existing DRL algorithms impractical for the data centers under investigation. In this paper, we carefully address the aforementioned challenges and propose deep learning-based algorithms to achieve more accurate predictions and effectively handle enormous action spaces, ultimately leading to superior optimization performance.

3 System model

The IoT-enabled data center and the deep learning-based PUE optimization framework are illustrated in Fig. 1. The left side of Fig. 1 illustrates the IoT-enabled data center, where servers in the server room generate heat during operation. To dissipate this heat, a chiller cooling system is employed to provide the necessary cooling capacity. The cooling capacity is achieved through the collaboration of multiple devices. Specifically, the chiller generates cold water which is then pumped by a chilled water pump to the server room and the cold water tank, for the purpose of heat dissipation and cold water storage, respectively. The outlet water from the server room is subsequently pumped back to the chiller, where the temperature is higher than the inlet water due to heat exchange with the servers. Subsequently, the chiller cools the outlet water using a plate heat exchanger and a cooling tower. In this process, a cooling water pump is responsible for pumping the cooling water, and the cooled outlet water is returned to the chiller for circulation. Utilizing IoT techniques, a multitude of sensor nodes, such as temperature sensors, humidity sensors, and flowmeter sensors, are deployed across these devices. Through these sensors, the global state of the data center can be monitored in real time. The data sensed by the IoT devices is then collected either wirelessly or through wired links and subsequently processed by the gateway. The processed data is then uploaded to the data center infrastructure management (DCIM) system and stored in a database. By retrieving the data from the database, we construct a deep learning-based prediction network. Additionally, we propose a DRL-based optimization algorithm that interacts with the prediction network. The optimized strategies resulting from this algorithm are subsequently sent to the chiller cooling system via the DCIM. The IoT devices serve as a vital link between the physical data center and the digital optimization framework, as depicted on the left and right sides of Fig. 1.

In the data center, the total power consumption can be categorized into the following components:

$$P_{\text{total}} = P_{\text{supply}} + P_{\text{IT}} + P_{\text{cooling}} + P_{\text{CRAC}} + P_{\text{others}}, \quad (1)$$

where P_{supply}, P_{IT}, P_{cooling}, P_{CRAC}, and P_{others} represent the power consumption of the power supply, information technology (IT) devices, chiller cooling system, precision air conditioners in server rooms, and other miscellaneous factors such as lighting, security, and office equipment, respectively.

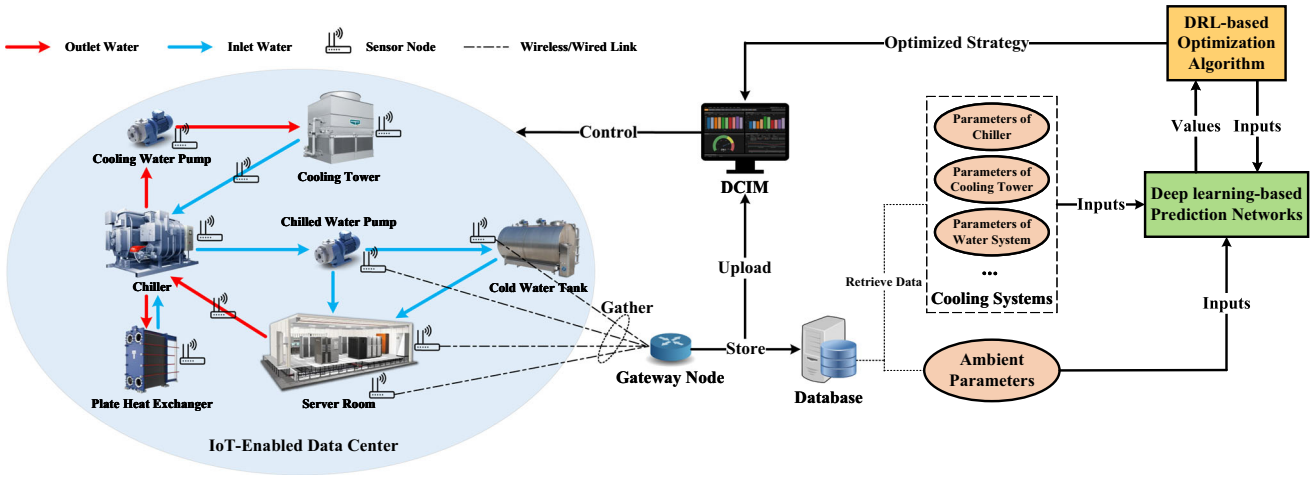

Fig. 1 A deep learning-based PUE optimization framework for IoT-enabled data center

Moreover, as illustrated in Fig. 1, the power consumption of the chiller cooling system can be further subdivided into the following components:

$$P_{\text{cooling}} = P_{\text{CH}} + P_{\text{CWP}} + P_{\text{CHP}} + P_{\text{CT}} + P_{\text{others}} + P_{\text{loss}}, \tag{2}$$

where P_{CH}, P_{CWP}, P_{CHP}, P_{CT}, P_{others}, and P_{loss} represent the power consumption of the chiller, chilled water pump, cooling water pump, cooling tower, other devices in the chiller cooling system, and power loss due to transmission loss and heat dissipation, respectively.

To compare the energy efficiency of different data centers, it is necessary to define common metrics. One widely recognized and utilized metric is the PUE. PUE is defined as the ratio of the total energy consumption of the data center to the energy consumption of IT devices over a specific time period [28], as shown below:

$$\text{PUE} = \frac{E_{\text{total}}}{E_{\text{IT}}}, \tag{3}$$

where E_{IT} and E_{total} represent the summation of P_{IT} and P_{total} over the specified time period, respectively. As indicated in Eqs. 1 and 2, PUE ≥ 1 due to the energy consumption of other factors. A lower PUE value indicates a higher fraction of power being delivered to the IT devices and signifies greater energy efficiency in the data center.

3.1 Problem formulation

PUE optimization in data centers is a continuous-time optimization problem. To discretize a period of time into T time slots, we propose the following general problem formulation:

$$\min_{\mathbf{X_c}} \quad \text{PUE}\left(\mathbf{X}_c, \mathbf{X}_s, \mathbf{X}_e\right) \tag{4a}$$

$$\text{s.t.} \quad C_i\left(\mathbf{X}_c, \mathbf{X}_s, \mathbf{X}_e\right) \leq 0, \ \forall i \in \{1, 2, \cdots, I\}, \tag{4b}$$

$$S_j\left(\mathbf{X}_c, \mathbf{X}_s, \mathbf{X}_e\right) \leq 0, \ \forall j \in \{1, 2, \cdots, J\}, \tag{4c}$$

where the matrices $\mathbf{X}_c \in \mathbb{R}^{d_c \times T}$, $\mathbf{X}_s \in \mathbb{R}^{d_s \times T}$, and $\mathbf{X}_e \in \mathbb{R}^{d_e \times T}$ represent collections of vectors for all time slots, denoted as $\mathbf{X}_i = [\mathbf{x}_i(1), \mathbf{x}_i(2), \cdots, \mathbf{x}_i(T)]$ for $i \in \{c, s, e\}$. Specifically, $\mathbf{X}_c$ contains the controllable parameters of chiller cooling system devices, $\mathbf{X}_s$ contains the status (uncontrollable) parameters of devices, and $\mathbf{X}_e$ contains the environmental parameters of the data center. Each vector $\mathbf{x}_c(t) \in \mathbb{R}^{d_c}$, $\mathbf{x}_s(t) \in \mathbb{R}^{d_s}$, and $\mathbf{x}_e(t) \in \mathbb{R}^{d_e}$ corresponds to the respective parameters in time slot t. The constraints (4b) correspond to the physical limitations imposed by the data center devices, while constraints (4c) represent the safety restrictions associated with these devices. The variables I and J represent the number of physical and safety constraints, respectively. Problem (4) aims to minimize the PUE by optimizing the controllable parameters of chiller cooling system devices $\mathbf{x}_c(t)$ across all time slots $t = \{1, \cdots, T\}$ while satisfying the physical and safety constraints of data center devices throughout.

Given that the primary emphasis of this paper is the optimization of the chiller cooling system, the objective of minimizing the PUE is tantamount to minimizing the energy consumption of the cooling system, as defined in Eq. 3. Consequently, we can redefine problem (4) as follows:

$$\min_{\mathbf{X_c}} \quad \sum_{t=0}^{T} E_{\text{cooling}}\left(\mathbf{X}_c, \mathbf{X}_s, \mathbf{X}_e\right) \tag{5a}$$

$$\text{s.t.} \quad \text{(4b), (4c).} \tag{5b}$$

In fact, problem (5) can be regarded as a long-term energy consumption optimization problem. Furthermore, we specify the controllable parameters and safety constraints adopted in this paper, resulting in the following reformulation of problem (5):

$$\min_{\mathbf{X_c}} \quad \lim_{T\to\infty} \sum_{t=0}^{T} \frac{E_{\text{cooling}}(\mathbf{X}_c,\mathbf{X}_s,\mathbf{X}_e)}{T} \tag{6a}$$

$$\text{s.t.} \quad \text{ChWOT}^l \le \text{ChWOT}_i(t) \le \text{ChWOT}^u, \tag{6b}$$

$$\text{ChWOP}^l \le \text{ChWOP}_i(t) \le \text{ChWOP}^u, \tag{6c}$$

$$\text{CoWOT}^l \le \text{CoWOT}_i(t) \le \text{CoWOT}^u, \tag{6d}$$

$$\text{CoWOP}^l \le \text{CoWOP}_i(t) \le \text{CoWOP}^u, \tag{6e}$$

$$\text{EOT}^l \le \text{EOT}_i(t) \le \text{EOT}^u, \tag{6f}$$

$$t \in \{1, 2, \cdots, T\}, i = 1 \text{ or } 2, \tag{6g}$$

where the controllable variable $\mathbf{X}_c$ encompasses the parameters mentioned below: chilled water outlet temperature (ChWOT), chilled water outlet pressure (ChWOP), cooling water outlet temperature (CoWOT), cooling water outlet pressure (CoWOP), and evaporator outlet temperature (EOT). Since the cooling system comprises two chillers, the subscript i indicates that the variables correspond to chiller i, resulting in a total of 10 controllable variables considered in this study. Furthermore, each variable is discrete and can take on 21 different values. This leads to a total of 21^{10} possible combinations of solutions, which represents an extremely large number and poses a significant challenge for resolution.

To address this problem, we first employ the TPN $E_{\text{cooling}} = \mathcal{M}(\mathbf{X}_c, \mathbf{X}_s, \mathbf{X}_e)$ to predict the energy consumption of the chiller cooling systems in Section 4, which serves as the basis for the subsequent optimization using DRL in Section 5.

4 Transformer-based energy consumption prediction

In this section, we present the problem formulation for predicting the energy consumption of a chiller cooling system and provide a detailed introduction to the architecture of the proposed TPN.

4.1 Energy consumption prediction

We frame the prediction of chiller cooling system energy consumption as a temporal sequence prediction problem, considering the time-dependent nature of temperature, humidity, and the cooling system itself. The predicted output of the network is denoted as $\mathbf{y} = [y(1), \cdots, y(T)]$, where $y(t)$ represents the energy consumption of the chiller cooling system at time slot t. The input to the prediction network is represented as $\mathbf{X} = [\mathbf{x}(1), \cdots, \mathbf{x}(T)]$, where $\mathbf{x}(t) \in \mathbb{R}^{d_m}$, $d_m = d_c + d_s + d_e$ represents the collection of cooling system parameters and environmental parameters at time slot t:

$$\mathbf{x}(t) = [\mathbf{x}_c(t), \mathbf{x}_s(t), \mathbf{x}_e(t)]^T. \tag{7}$$

Given the historical inputs $\mathbf{X}_{1:t}$ up to time slot t, the energy consumption prediction model $\mathcal{M}$ aims to learn the energy consumption $\hat{y}(t+1)$ of the cooling system at the next time step. Therefore, the energy consumption prediction problem can be formulated as:

$$\hat{y}(t+1) = \mathcal{M}(\mathbf{X}_{1:t}). \tag{8}$$

4.2 Transformer-based prediction network

In this subsection, we present a comprehensive overview of the TPN architecture, specifically designed for estimating the energy consumption of chiller cooling systems. As depicted in Fig. 2, the input matrix $\mathbf{X}$[1] undergoes positional encoding to integrate positional information. It then progresses through the transformer layer, which comprises multi-head attention, layer normalization, and feed-forward operations to capture temporal correlations. Ultimately, the processed data is input into the prediction layer for forecasting the energy consumption of the cooling systems. More comprehensive information regarding this network is provided in the following subsections.

4.2.1 Positional encoding layer

In recurrent models, the previous hidden state serves as the input at the current time step, preserving the order of the sequence. In convolutional models, some positional information may be lost due to the convolutional kernel operation, although positional relationships between extracted features are still preserved. However, in the TPN, there is no inherent operation that provides positional information for sequences. In light of that, we introduce a positional encoding layer to inject positional information into the input sequence. There are various methods for positional encoding, including learned and fixed approaches. In this paper, we employ a fixed positional encoding method [29], which enables the model to effectively capture relative positions. We denote the positional encoding matrix as $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{T\times d_m}$, aligned with the dimensionality of the input matrix. The (i, j) element of $\mathbf{E}_{\text{pos}}$ is defined as:

$$\mathbf{E}_{\text{pos}}(i, j) = \begin{cases} \sin\left(i/10000^{\frac{j}{d_m}}\right), & \text{if } j \text{ is even,} \\ \cos\left(i/10000^{\frac{j}{d_m}}\right), & \text{if } j \text{ is odd,} \end{cases} \tag{9}$$

[1] In this subsection, for simplicity, we represent $\mathbf{X}_{1:t}$ as $\mathbf{X}$.

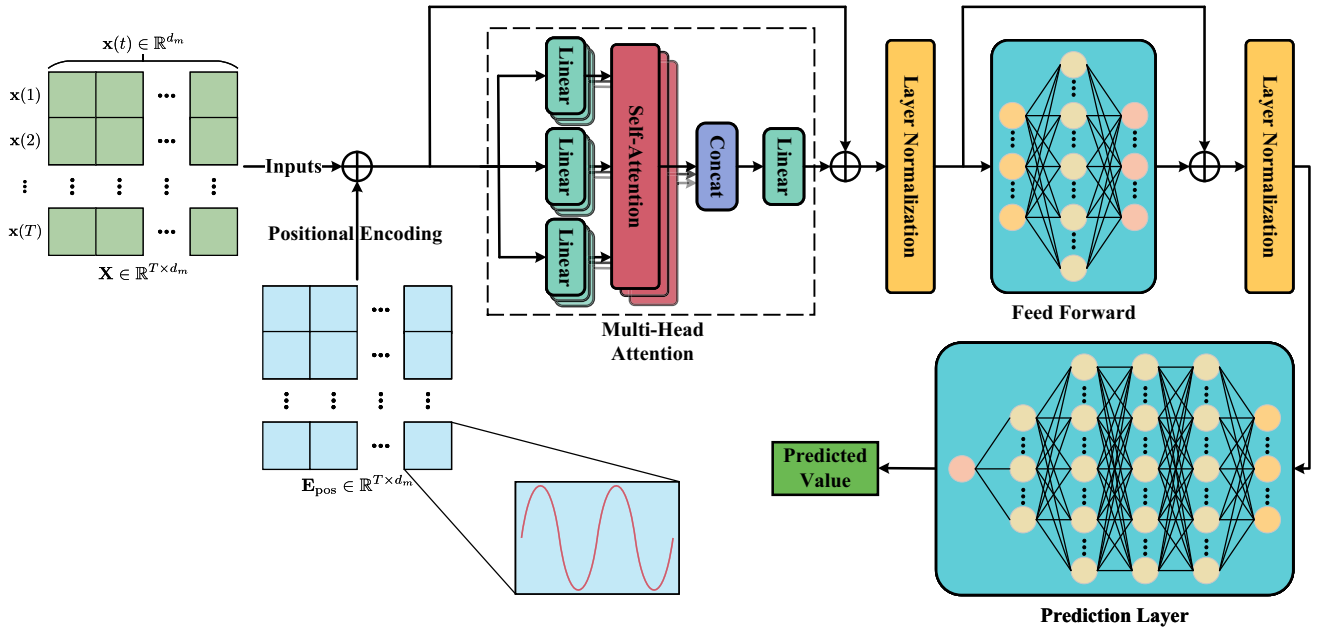

Fig. 2 The network architecture of TPN

where i represents the position, and j denotes the dimension index. By incorporating the positional encoding into the input matrix, we attain the positional information, and the output of the positional encoding layer is given by:

$$\mathbf{X}_p = \mathbf{X} + \mathbf{E}_{\text{pos}}. \tag{10}$$

4.2.2 Transformer layer

The transformer layer plays a crucial role in constructing and capturing temporal correlations within the input sequence. It achieves this by incorporating multiple attention heads, residual connections, layer normalization, and a feed-forward network. The positional encoded input $\mathbf{X}_p$ serves as the input to the transformer layer, and the resulting output is a feature matrix denoted as $\mathbf{X}_t$, which represents the extracted temporal correlations.

1) Multi-head attention

The attention mechanism, initially introduced in machine translation by Bahdanau et al. [30], has found widespread applications in neural networks across various fields. The underlying purpose of attention is to enable the model to emphasize the most relevant parts of the inputs by assigning higher weights (attentions) to these features. This approach offers various benefits, such as reduced complexity, parallelized computation, and the ability to capture long-range dependencies.

In the realm of attention mechanisms, there exist numerous variants, as surveyed by Chaudhari et al. [31]. In this paper, we adopt the self-attention mechanism to explore temporal correlations within the sequence. The self-attention mechanism utilizes query matrix $\mathbf{Q} \in \mathbb{R}^{T \times d_t}$, key matrix $\mathbf{K} \in \mathbb{R}^{T \times d_t}$, and value matrix $\mathbf{V} \in \mathbb{R}^{T \times d_t}$ derived from the sequence itself, as shown below:

$$\mathbf{Q} = \mathbf{X}_p \mathbf{W}^Q, \ \mathbf{K} = \mathbf{X}_p \mathbf{W}^K, \ \mathbf{V} = \mathbf{X}_p \mathbf{W}^V, \tag{11}$$

where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d_m \times d_t}$ are the trainable parameter matrices, and d_t represents the length of the second dimension of the query, key, and value matrices. Subsequently, the self-attention is computed in the following manner:

$$\mathcal{T}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_t}}\right)\mathbf{V}, \tag{12}$$

where $\mathbf{Q}\mathbf{K}^T$ denotes the attention score, and the scaling factor $1/\sqrt{d_t}$ is incorporated to mitigate the gradient vanishing issue in the softmax function caused by large dot product values. Besides, the softmax function is employed to normalize the attention scores for each time slot.

Through the integration of self-attention into the input sequence, we can identify correlations between any two time slots and assign varying attention weights to highlight the most pertinent parts. While a single self-attention mechanism captures relationships within a single degree, a natural approach to capturing correlations across multiple degrees is the simultaneous use of multiple self-attentions, known

as multi-head attention. This allows the model to concentrate on correlations from different representation subspaces concurrently. Each multi-head attention block comprises h self-attention blocks, defined as follows:

$$\mathcal{T}_M(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_h]\, W^O, \tag{13a}$$

$$\text{where } \mathcal{T}_i = \mathcal{T}\left(\mathbf{X}_p\mathbf{W}_i^Q, \mathbf{X}_p\mathbf{W}_i^K, \mathbf{X}_p\mathbf{W}_i^{\mathbf{V}}\right), \tag{13b}$$

where $\mathcal{T}_i$ represents the i-th self-attention block (head) within the multi-head attention mechanism. The parameter matrices $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V \in \mathbb{R}^{d_m \times d_t}$ are learnable matrices specific to the i-th head. The outputs of the h heads are concatenated and passed through a linear layer represented by the learnable parameter matrix $\mathbf{W}^O \in \mathbb{R}^{hd_t \times d_m}$ to yield the ultimate output $\mathbf{X}_z$ of the multi-head attention block.

2) Residual connection, layer normalization and feed forward

Figure 2 illustrates the presence of residual connections, layer normalization, and feed-forward operations following the multi-head attention in the transformer layer. Residual connections are initially employed in deep learning networks for image recognition [32] to mitigate the issue of vanishing and exploding gradients during the training of deep neural networks. These connections introduce identical shortcut connections that bypass one or more layers, allowing for direct information propagation and enabling the network to more effectively learn residual information. Additionally, layer normalization [33] is utilized to alleviate the problem of internal covariate shift during training. A position-wise feed-forward network is also applied in this context to enhance the model's representation and learning ability. This network consists of a two-layer fully connected architecture with a ReLU activation function between the layers, which is applied separately and identically to each position.

Specifically, the output of the multi-head attention $\mathbf{X}_z$ undergoes residual connection and layer normalization as follows:

$$\mathbf{X}_z = \mathrm{LN}\left(\mathbf{X}_p + \mathbf{X}_z\right), \tag{14}$$

where $\mathrm{LN}(\cdot)$ represents the layer normalization operation. Then, it is fed into the feed-forward network:

$$\mathbf{X}_f = \mathrm{ReLU}\left(\mathbf{X}_z\mathbf{W}_f^1 + \mathbf{b}_f^1\right)\mathbf{W}_f^2 + \mathbf{b}_f^2, \tag{15}$$

where $\mathbf{W}_f^1 \in \mathbb{R}^{d_m \times d_f}$, $\mathbf{b}_f^1 \in \mathbb{R}^{d_f}$, $\mathbf{W}_f^2 \in \mathbb{R}^{d_f \times d_m}$, and $\mathbf{b}_f^2 \in \mathbb{R}^{d_m}$ represent the learnable parameter matrix and vector for the first and second layers of the feed-forward network, respectively. In order to capture more complex features, it is necessary that $d_f > d_m$. The activation function used in the feed-forward network is the rectified linear unit (ReLU), defined as $\mathrm{ReLU}(x) = \max(0, x)$, and it operates on a position-wise basis. Finally, residual connection and layer normalization are applied once again:

$$\mathbf{X}_t = \mathrm{LN}\left(\mathbf{X}_z + \mathbf{X}_f\right). \tag{16}$$

4.2.3 Prediction layer

The prediction layer aims to transform the extracted correlations and features obtained from the transformer layer into predicted energy consumption values. This layer is implemented as a multi-layer perceptron consisting of multiple fully connected layers, as depicted below:

$$\mathbf{h}^{(l+1)} = f_\sigma^{(l)}\left(\mathbf{W}^{(l)}\mathbf{h}^{(l)} + \mathbf{b}^{(l)}\right),\ \forall l \in \{0, 1, \cdots, L-1\}, \tag{17}$$

where $\mathbf{h}^{(l)} \in \mathbb{R}^{d^{(l)}}$ represents the input of the l-th layer, while $\mathbf{h}^{(l+1)} \in \mathbb{R}^{d^{(l+1)}}$ represents the output. The weight matrix and bias of the l-th layer are denoted as $\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{d^{(l+1)}}$, respectively. The activation function $f_\sigma^{(l)}(x)$ is adopted as the ReLU. Notably, the input of the first layer is the output of the transformer layer, represented as $\mathbf{h}^{(0)} = \mathbf{X}_t$, while the output of the last layer corresponds to the predicted energy consumption, denoted as $\mathbf{h}^{(L)} = \hat{y}(t+1)$.

5 DRL-based energy consumption optimization

In this section, we introduce a modeling approach for the long-term energy consumption optimization problem (6) by formulating it as an MDP with an enormous action space. To address this challenge, we propose a solution called the BD3QN. The BD3QN aims to compress the action space by dividing it into multiple sub-action spaces. This approach enables effective handling of the enormous action space, thereby enhancing overall performance.

5.1 Markov decision process modeling

In the context of reinforcement learning (RL), the agent's decision-making process involves selecting actions based on a policy. Subsequently, the environment furnishes the agent with a reward determined by the selected action and the current state. Through a series of interactions between the agent and the environment, RL's primary goal is to acquire an optimal policy that maximizes the cumulative reward. This long-term reward is especially well-suited for addressing our specific long-term optimization problem (6). To represent

this procedure, it is customary to employ an MDP framework. Thus, we initially model problem (6) as an MDP, which can be described by a five-tuple $< \mathcal{S}, \mathcal{A}, P, r, \gamma >$. Here, $\mathcal{S}, \mathcal{A}, P, R, \gamma$ denote the state space, action space, state transition probability function, reward function, and discount factor, respectively. The subsequent subsections provide a detailed presentation of the MDP modeling.

State The state represents the information obtained by the agent from the environment, serving as a real-time representation of the physical environment and being influenced by the agent's actions. In this paper, we delineate the chiller cooling system's state as follows:

$$s_t = \left\{ \mathbf{x}_s(t), \mathbf{x}_e(t) \right\}, \tag{18}$$

where the state s_t of the cooling system at time slot t encompasses both the operational parameters $\mathbf{x}_s(t)$ of the cooling system and the environmental parameters $\mathbf{x}_e(t)$.

Action The agent's goal is to reduce the energy consumption of the chiller cooling system through adjusting the system's controllable device parameters, which are regarded as the agent's actions. Consequently, we specify the action space as follows:

$$a_t = \{\mathbf{x}_c(t)\}, \tag{19}$$

where $\mathbf{x}_c(t)$ represents the 10 controllable variables utilized in problem (6). Thus, the action space is a 10-dimensional space, with each dimension comprising 21 discrete variables. As a result, the action space encompasses a total of $21^{10} \approx 1.7 \times 10^{13}$ actions, indicating its extremely large size.

State transition Given the current state s_t, the agent selects action a_t, and subsequently, the environment returns rewards and transitions to the next state s_{t+1} based on the stochastic state transition probability function $P(s_{t+1}|s_t, a_t)$. This function cannot be obtained in advance and needs to be approximated through multiple interactions between the agent and the environment, involving a sampling process.

Reward The reward received by the agent at state s_t when taking action a_t is represented as r_t. To diminish the long-term energy consumption of the chiller cooling system and, subsequently, lower the PUE, we establish the reward function as the negative energy consumption:

$$r_t = -E_{\text{cooling}}\left(\mathbf{X}_c, \mathbf{X}_s, \mathbf{X}_e\right), \tag{20}$$

where $E_{\text{cooling}}\left(\mathbf{X}_c, \mathbf{X}_s, \mathbf{X}_e\right)$ is obtained through the TPN $\mathcal{M}$ introduced in Section 4.

5.2 Dueling double deep Q-network

The dueling double deep Q-network (D3QN) is an RL algorithm that combines the dueling architecture and the double deep Q-network (DQN) approach [23]. By decomposing the action value function into a state value function and an advantage function, dueling DQN effectively handles the differences between different actions, reduces the learning complexity, and achieves significant performance improvements. Moreover, the adoption of double DQN helps to mitigate the overestimation of action values.

5.2.1 Deep Q-network

RL aims to maximize the long-term reward through interactions between an agent and its environment. The reward is composed of both immediate and future rewards, discounted by a factor γ as defined by the following equation:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}. \tag{21}$$

In pursuit of maximizing the long-term reward, the optimal action must be selected for each state. To evaluate the value of action a_t at state s_t, we introduce the action value function $Q(s_t, a_t)$ as follows:

$$Q(s_t, a_t) = \mathbb{E}\left[G_t | s = s_t, a = a_t\right]. \tag{22}$$

In the real world, directly obtaining $Q(s_t, a_t)$ is not feasible, but we can approximate it using the temporal difference (TD) method. Through iteratively interacting with the environment, $Q(s_t, a_t)$ is iteratively approximated according to the following update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[y^{\text{TD}} - Q(s_t, a_t) \right], \tag{23}$$

where α represents the step size, and the TD target is denoted as $y^{\text{TD}} = r_t + \gamma \max_a Q(s_{t+1}, a)$. During the interaction process, the action a is selected using an ϵ-greedy policy:

$$a = \begin{cases} \arg\max_{a'} Q(s, a'), & p = 1 - \epsilon, \\ \text{random } a \in \mathcal{A}, & p = \epsilon, \end{cases} \tag{24}$$

where p represents the probability of executing the action.

The DQN was introduced as a solution to the challenges encountered when dealing with large or continuous state and action spaces [22]. Instead of relying on a traditional Q-table $Q(s, a)$ as in Eq. 23, the DQN employs a deep neural network called the Q-network, denoted as $Q_{\boldsymbol{\theta}}(s, a)$. Furthermore, to mitigate the issue of unstable training, a target Q-network, represented by $Q_{\boldsymbol{\theta}^-}(s, a)$, is introduced. This target network

serves to stabilize the training process by maintaining fixed training targets for a specific duration. The parameters of the target Q-network are updated every C steps using the values from the primary network $Q_{\boldsymbol{\theta}}(s, a)$. As a result, the TD target of the DQN differs from the one defined in Eq. 23 and can be expressed as follows:

$$y^{\mathrm{DQN}} = r_t + \gamma \max_a Q_{\boldsymbol{\theta}^-}(s_{t+1}, a). \tag{25}$$

During the training stage, a time-dependent sequence of experiences (s, a, r, s') is generated, which hinders the effective training of the Q-network. Moreover, reusing experiences to improve efficiency is desired. To address these issues, experience replay is introduced. This involves storing the sampled experiences in a replay buffer denoted as $\mathcal{D}$, from which a batch of samples is randomly selected for training the Q-network. The loss function employed during training is defined as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(y^{\mathrm{DQN}} - Q_{\boldsymbol{\theta}}(s_t, a_t)\right)^2\right]. \tag{26}$$

5.2.2 D3QN

The estimation of the Q-network involves both positive and negative estimation errors. However, in the case of DQN, the same Q-network is used for action selection and evaluation, by selecting actions with maximum Q-values, as shown in Eqs. 23 and 25. This approach leads to the accumulation of positive estimation errors. To address this overestimation problem in DQN, the double DQN algorithm [24] is proposed. Double DQN tackles the issue by decomposing the max operation in the TD target into action selection and action evaluation. In double DQN, the two Q-networks used in DQN serve as a natural choice without requiring additional networks. Action selection is performed by $Q_{\boldsymbol{\theta}}(s, a)$ using an ϵ-greedy policy, while $Q_{\boldsymbol{\theta}^-}(s, a)$ is used to evaluate the action. The update process of double DQN remains the same as DQN, but the TD target y^{DQN} is replaced with:

$$y^{\text{double DQN}} = r_t + \gamma Q_{\boldsymbol{\theta}^-}\left(s_{t+1}, \arg\max_a Q_{\boldsymbol{\theta}}(s_{t+1}, a)\right). \tag{27}$$

Building upon double DQN, the D3QN approach [23] proposes to separate the action value function $Q(s, a)$ into the state value function $V(s)$ and the advantage function $A(s, a)$, as shown below:

$$Q_{\boldsymbol{\theta}}(s, a) = V_{\boldsymbol{\theta}}(s) + A_{\boldsymbol{\theta}}(s, a) - \frac{1}{|\mathcal{A}(s)|}\sum_{a'\in\mathcal{A}(s)} A_{\boldsymbol{\theta}}(s, a'), \tag{28}$$

where $\mathcal{A}(s)$ represents the set of available actions at state s. Considering the modification in Eq. 28, the architecture of the Q-network is also altered. The original Q-network is divided into the state value sub-network $V_{\boldsymbol{\theta}}(s)$ and the advantage sub-network $A_{\boldsymbol{\theta}}(s, a)$, which are then combined to obtain the action value function $Q_{\boldsymbol{\theta}}(s, a)$ according to Eq. 28. Through these operations, D3QN can effectively focus on the value of actions or states and handle larger action spaces more efficiently.

5.3 Branching action space for compression

While traditional DRL algorithms demonstrate commendable performance, they encounter difficulties in handling large action spaces (encompassing approximately 1.7×10^{13} actions in this paper) for several reasons. First, the extremely large size of the action space makes it difficult for them to directly learn and explore effective strategies. Second, these algorithms often require an extensive number of samples for learning and exploration, leading to low sample-efficiency and time consumption. Third, the output dimensions of Q-networks in these traditional methods correspond to the total possible actions, resulting in computationally intensive large-scale neural networks. To tackle this challenge, we propose a solution that involves branching the action space into multiple sub-action spaces, aiming to achieve compression while preserving a certain level of interdependence among the sub-actions [34].

In this paper, we extend the D3QN framework by introducing action branching. Specifically, we branch the advantage sub-network into multiple sub-advantage networks, while the action-independent state value functions are shared across all sub-action spaces. This modified framework is referred to as BD3QN and is depicted in Fig. 3. Significantly, these sub-actions are not entirely independent, as they rely on the shared representation utilized by the sub-advantage networks. This level of interdependence is crucial for coordinating the sub-actions [34] to achieve better performance. It has been demonstrated that the absence of such interdependence would deteriorate the performance [34]. By combining the sub-advantage networks and the state value sub-network, we derive the action value function for each sub-action. The formal description and definition of BD3QN are presented as follows.

We first decompose the action space $\mathcal{A}$ into D sub-action spaces $\{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_D\}$, then the action value function for sub-action $a_d \in \mathcal{A}_d$ at state s can be represented as:

$$Q_{\boldsymbol{\theta}_d}(s, a_d) = V_{\boldsymbol{\theta}}(s) + A_{\boldsymbol{\theta}_d}(s, a_d) - \frac{1}{|\mathcal{A}_d|}\sum_{a'_d\in\mathcal{A}_d} A_{\boldsymbol{\theta}_d}(s, a'_d), \tag{29}$$

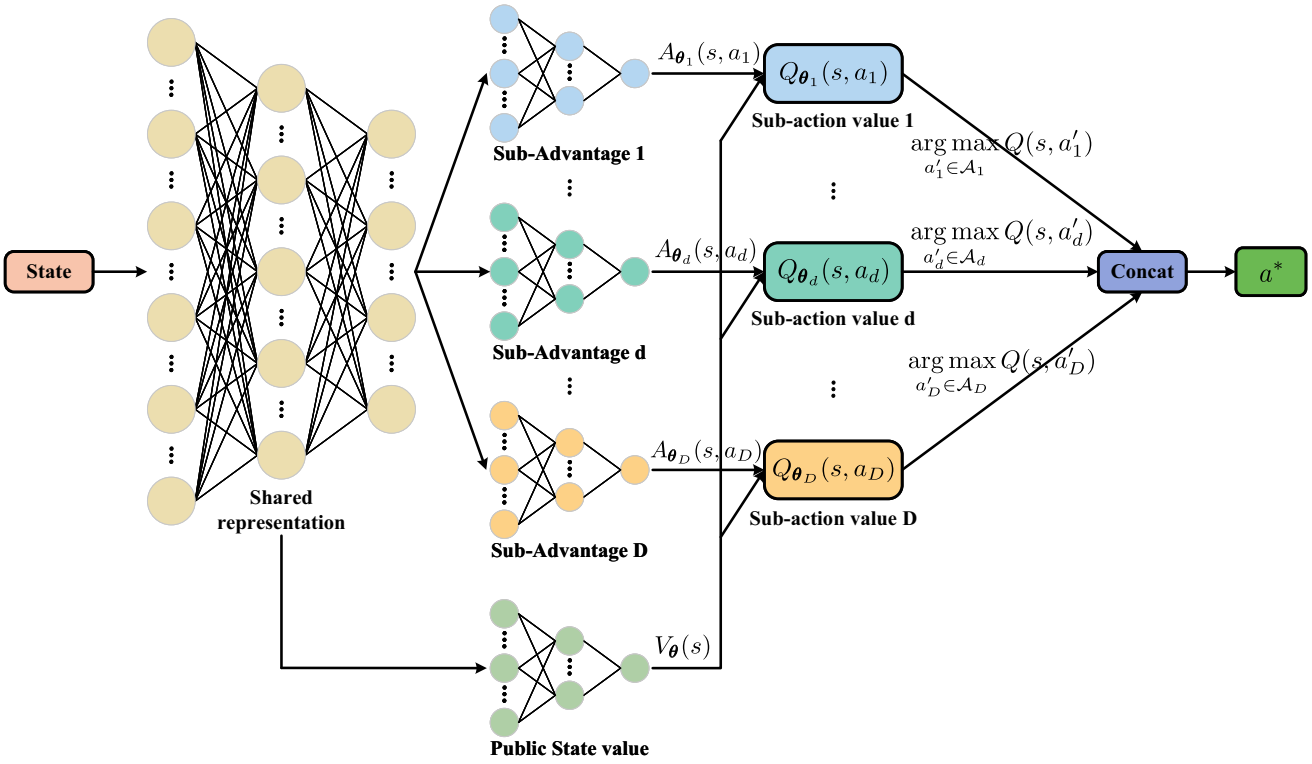

Fig. 3 The network architecture of BD3QN

where the state s serves as a shared component across all sub-actions. The parameter $\boldsymbol{\theta}_d$ denotes the parameters of the sub-action value function $Q_{\boldsymbol{\theta}_d}(s, a_d)$. Furthermore, it should be noted that there exists partial parameter overlap among $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_d$.

In BD3QN, the selection of actions is contingent upon the selection of sub-actions, as shown below:

$$a = \{a_1, a_2, \cdots, a_D\}, \tag{30}$$

where the sub-action a_d is selected from the d-th sub-action space using an ϵ-greedy policy:

$$a_d = \begin{cases} \arg\max_{a'_d \in \mathcal{A}_d} Q(s, a'_d), & p = 1 - \epsilon, \\ \text{random } a_d \in \mathcal{A}_d, & p = \epsilon. \end{cases} \tag{31}$$

To address the overestimation problem and consider multiple sub-action spaces, BD3QN adopts a TD target similar to D3QN. The TD target for the d-th sub-action space is defined as follows:

$$y_d = r_t + \gamma\, Q_{\boldsymbol{\theta}_d^-}\left(s_{t+1}, \arg\max_{a_d \in \mathcal{A}_d} Q_{\boldsymbol{\theta}_d}(s_{t+1}, a_d)\right), \tag{32}$$

Consequently, the loss function of BD3QN is defined as the mean squared error across the sub-action spaces:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\frac{1}{D}\sum_d \left(y_d - Q_{\boldsymbol{\theta}_d}(s_t, a_d)\right)^2\right], \tag{33}$$

where a denotes the multidimensional action.

Considering the enormous action space in our problem, there are numerous experiences stored in the replay buffer. However, uniformly sampling the data for training may lead to low efficiency. To address this issue, it is crucial to prioritize experiences with higher TD errors, as they can accelerate the training process. Therefore, this paper adopts prioritized experience replay [35], which improves sample efficiency and results in a better policy. The importance of experience (s, a, r, s') in prioritized experience replay is evaluated based on the TD error. Specifically, the TD error is the sum of TD errors across sub-actions in BD3QN:

$$\delta(s, a, r, s') = \sum_d \left(y_d - Q_{\boldsymbol{\theta}_d}(s, a_d)\right). \tag{34}$$

To determine the probability of replaying experience (s, a, r, s'), a formula is defined as follows:

$$P(s, a, r, s') = \frac{\left(|\delta(s, a, r, s')| + \epsilon\right)^\mu}{\sum_{(s,a,r,s')\in\mathcal{D}} \left(|\delta(s, a, r, s')| + \epsilon\right)^\mu}, \tag{35}$$

where μ denotes the prioritization exponent, and ϵ is a small positive constant that prevents experiences with zero probability of being replayed. However, prioritized experience replay may introduce bias into the training process due to non-uniform sampling. To overcome this issue, importance sampling weights are introduced:

$$w(s,a,r,s') = \left(\frac{1}{|\mathcal{D}|}\cdot\frac{1}{P(s,a,r,s')}\right)^{\beta}, \tag{36}$$

where $|\mathcal{D}|$ indicates the number of samples, and β is the importance sampling exponent. Finally, the loss function of BD3QN is modified as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[w(s,a,r,s')\cdot\frac{1}{D}\sum_{d}\left(y_d - Q_{\boldsymbol{\theta}_d}(s_t,a_d)\right)^2\right]. \tag{37}$$

The comprehensive training procedure is outlined in Algorithm 1.

By leveraging the BD3QN architecture, the exponential growth of the number of actions can be effectively reduced to a linear scale. In an N-dimensional action space with n_d discrete sub-actions for the d-th dimension, the total number of actions is given by $\prod_{d=1}^{N} n_d$. Consequently, achieving convergence and learning the optimal policy becomes a challenging task. However, the BD3QN approach addresses this challenge by branching the N-dimensional action space into N sub-action spaces. As a result, the total number of actions becomes the summation of the sub-actions, i.e., $\sum_{d=1}^{N} n_d$. This transformation significantly enhances the feasibility of addressing the problem posed by large-scale action spaces.

Algorithm 1 BD3QN-based energy consumption optimization for Chiller cooling system.

Input : Inital sate s_0.
Output : Optimal action $a^*(t)$.
Initialization: Establish two BD3QN Q-networks, $Q_{\boldsymbol{\theta}}$ and $Q_{\boldsymbol{\theta}^-}$, with the same initialized parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^-$. Initialize the prioritized experience replay;
for *episode* = 1 : N **do**
  Reset the environment and obtain the initial state s_0;
  for *time step* $t = 1 : S_T$ **do**
    The agent observes the environmental state s_t and selects the sub-action a_d for each d-th sub-action space, $\forall d \in \{1, 2, \cdots, D\}$, based on Eq. 31. Then, the sub-actions are combined to obtain the overall action a_t;
    Perform action a_t and obtain the corresponding reward r_t using the TPN. Transition to the next state s_{t+1};
    Store the experience (s_t, a_t, r_t, s_{t+1}) in the prioritized experience replay buffer $\mathcal{D}$;
    Assign a prioritized weight to each experience according to Eq. 35 and sample a batch of experiences from $\mathcal{D}$ when the data volume reaches the minimum batch size;
    Perform gradient descent on the loss function (37) across the sampled experiences to update the parameters of the Q-network $\boldsymbol{\theta}$;
    Update the parameters of the target Q-network $\boldsymbol{\theta}^-$ with the parameters of $\boldsymbol{\theta}$ every C steps;
  end
end

6 Experiments

In this section, we thoroughly evaluate our proposed algorithms using a real-world dataset, comparing them with other benchmarks. We commence by introducing the dataset, detailing preprocessing steps, and elucidating our experimental setup. Subsequently, we provide a comprehensive presentation and detailed analysis of the experimental results.

6.1 Experimental setup

The experimental data were collected from a data center situated in Jinan, Shandong province, China, using various IoT devices such as temperature and humidity sensors, flowmeter sensors, and others. Data collection commenced on May 31, 2022, at 08:39 and concluded on December 8, 2022, at 04:43, with a sampling frequency of either one or two minutes. In order to enhance the raw data's quality and usability, preprocessing measures are undertaken to mitigate errors and biases in data analysis and modeling. Initially, we apply linear interpolation to standardize the time intervals, followed by Z-score normalization to transform the data. Recognizing that uninformative features have a negligible impact on predictions, we eliminate single-valued features, commonly referred to as zero variance features, to reduce the computational load during the training process. Additionally, we utilize the box-whisker plot method to identify outliers, thereby ameliorating the adverse effects of inconsistent data. It is noteworthy that this method is robust and does not necessitate data adherence to a specific statistical distribution. As a result of these preprocessing steps, the processed dataset now consists of 190 dimensions, incorporating 699 input features and one energy consumption label, with a total of 188,747 samples.

TPN employs a historical data input of $T = 8$ time slots. The prediction layer comprises three layers with 190, 64, 32, and 1 neurons, respectively. The batch size is set at 512, the learning rate is fixed at 3×10^{-4}, and the Adam optimizer with mean squared error (MSE) is employed. In BD3QN, the initial exploration ratio is 0.001, the minibatch size is 150, the minimum replay memory size is 300, the shared representation layer size is (1024, 256), and each branch layer size is 128. BD3QN is trained using Adam with a learning

Table 1 Simulation parameters

Parameters	Values	Parameters	Values
d_m	190	h	4
d_t	190	d_f	380
ChWOT^l	18 °C	ChWOT^u	28 °C
ChWOP^l	0 bar	ChWOP^u	4 bar
CoWOT^l	17 °C	CoWOT^u	29 °C
CoWOP^l	0 bar	CoWOP^u	6 bar
EOT^l	14 °C	EOT^u	20 °C
μ	0.6	β	0.4
ϵ	0.01	γ	0.9
C	20	S_T	40

rate of 2×10^{-3}. The remaining hyperparameters for both TPN and BD3QN, as well as the simulation parameters, are provided in Table 1.

For prediction, we select a set of classical prediction algorithms for comparison with our proposed TPN. These include linear regression (LR), support vector regression (SVR), residual network (ResNet), GRU [13], and long short-term memory (LSTM). For the evaluation of their performance, we employ three commonly used metrics: root mean square error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE), which are defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}, \tag{38}$$

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T}|\hat{y}_t - y_t|, \tag{39}$$

$$\text{MAPE} = \frac{1}{T}\sum_{t=1}^{T}\frac{|\hat{y}_t - y_t|}{y_t} \times 100\%. \tag{40}$$

For optimization, we evaluate the proposed BD3QN against the following DRL algorithms: DQN [22], Double DQN [24], Dueling DQN [23], and D3QN [23].

6.2 Experiment results

6.2.1 Prediction performance

We select a specific time period for evaluating the performance of our proposed TPN, spanning from September 10, 2022, at 00:00 to September 23, 2022, at 20:00. The comparison between the predicted energy consumption of the TPN and the real energy consumption is depicted in Fig. 4. Notably, the predicted results closely align with the real values, underscoring the effectiveness of the proposed TPN.

To provide a quantitative assessment, we compare our proposed algorithms with other prediction benchmarks using multiple metrics, namely RMSE, MAE, and MAPE. The detailed results are presented in Table 2. Our comparative analysis reveals that the TPN achieves the lowest RMSE of 0.98% and the lowest MAPE of 1.40%, both of which significantly outperform the other benchmarks. In particular, the TPN yields a 75.5% reduction in RMSE and a 32.4% decrease in MAPE compared to the next best algorithm, GRU. This advantage is particularly pronounced when contrasted with all other benchmarks.

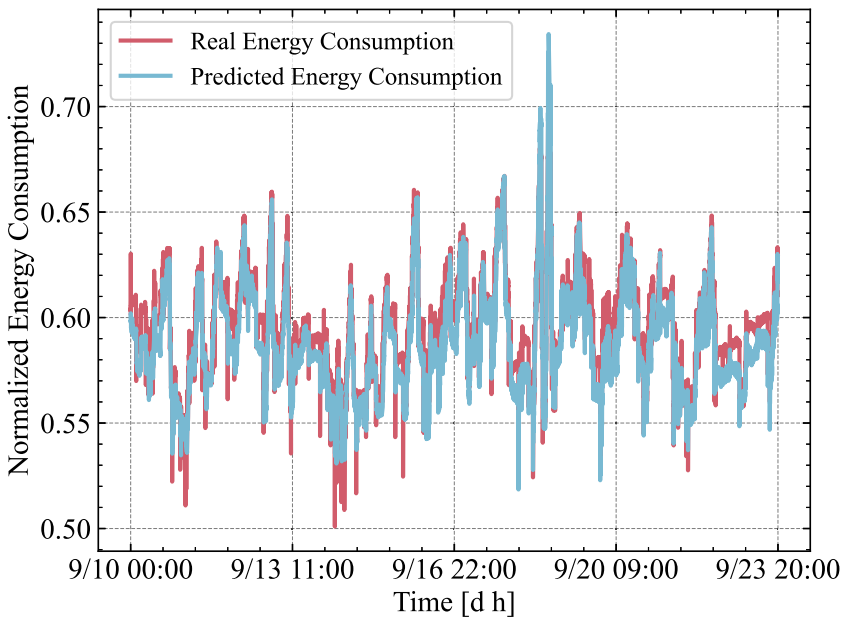

Fig. 4 Prediction results of TPN versus real values

Table 2 Performance comparison of various prediction algorithms (The best results are in **bold**)

Algorithm	RMSE	MAE	MAPE
LR	50.8%	3.32%	12.8%
SVR	16.4%	4.77%	15.85%
ResNet	6.33%	0.91%	3.19%
GRU	4.00%	**0.76**%	2.07%
LSTM	6.55%	1.49%	3.09%
TPN	**0.98**%	0.83%	**1.40**%

While the MAE of TPN is slightly less favorable than that of GRU, it remains advantageous in comparison to all the other algorithms. It is noteworthy to consider that RMSE primarily reflects the algorithm's ability to capture the overall data trend, whereas MAE assesses its ability to fit individual data points, and MAPE evaluates the relative error concerning the real values. Therefore, the observation of a slightly higher MAE alongside significantly improved RMSE and MAPE suggests that TPN excels in capturing the broader data trends for the majority of data points, although it may sacrifice some precision with respect to specific data points, possibly influenced by outliers.

In conclusion, the aforementioned experimental results decisively affirm the superiority of TPN over other algorithms. They underscore the effectiveness of its architectural design and its proficiency in capturing temporal dependencies within the dataset.

6.2.2 Optimization performance

As previously mentioned, the original action space consists of 10 dimensions, each encompassing 21 discrete values, resulting in a staggering total of 1.7×10^{13} possible actions. Such an expansive action space exceeds the manageable scope of classical DRL algorithms[2]. To facilitate feasible comparisons, we opt for a more tractable approach, reducing the action space to 5 dimensions, each with 6 discrete values, in our experiments involving DQN, double DQN, dueling DQN, and D3QN. The reward variations of these DRL algorithms during the training process are visualized in Fig. 5. The results clearly demonstrate that BD3QN converges to a value of -23 in approximately 1000 episodes, while the other DRL algorithms do not converge and consistently yield significantly lower rewards, around -24.5. Notably, BD3QN operates within an action space of size $6 \times 5 = 30$ whereas the other DRL algorithms grapple with a massive action space of $6^5 = 7776$. The significantly smaller action space in BD3QN facilitates faster convergence and leads to the acquisition of a more effective policy. Conversely, the impractically enormous action spaces of the other DRL algorithms render policy learning infeasible. Additionally, it is important to note that during the training phase, classical DRL algorithms consume considerably more time than BD3QN due to the increased complexity of their Q-networks resulting from the extremely larger action space.

We now turn our attention to the case of 10-dimensional actions, each consisting of 21 discrete values. The rewards' variations during the training process of BD3QN are depicted in Fig. 6. Notably, despite the enormous action space comprising approximately 1.7×10^{13} possible actions, BD3QN converges to a value of -19 in approximately 1000 episodes. This convergence speed is comparable to the 5-dimensional action case with 6 discrete values, but the results are markedly superior. The enhanced results can be ascribed to the increased number of parameters we managed and the greater precision we exercised in control. This achievement underscores BD3QN's exceptional capability to manage exceedingly large action spaces while maintaining advantages in both convergence and optimality. The efficacy of BD3QN can be attributed to the efficacy of its branching operation, which effectively reduces the number of actions from 1.7×10^{13} to a manageable 210.

We assess the performance of BD3QN during a specific time window, spanning from 23:00 to 05:00 on September 10, 2022. The comparative results of optimized energy consumption versus real values are illustrated in Fig. 7. Notably, the optimized and real energy consumption exhibit congruent trends, with a substantial reduction observed in the optimized energy consumption. The minimum reduction stands at 1.87%, while the maximum reaches 20.0%, and the mean reduction totals 12.1%. These findings underscore the efficacy of BD3QN in optimizing energy consumption.

Additionally, we conduct a comparison between optimized parameters and real parameters, as partially illustrated in Fig. 8[3], to offer insights for practical applications. It is discernible that the original real parameters remain largely unchanged, while the optimized parameters exhibit significant fluctuations over time. This observation underscores the necessity for dynamic changes in optimal parameters, as opposed to the static nature of suboptimal, unaltered parameters. This dynamic adaptability is a key reason why BD3QN surpasses human-operated systems where parameters often remain fixed for extended periods. In Fig. 8a and b, it is evident that the parameters of two chillers, despite being identical, follow distinct trends. This divergence arises from variations in chiller statuses and their mutual cooperation. Among these optimized parameters, no consistent trend or

[2] The Q-networks within them would necessitate vast computational resources, roughly around 60 TB in practice, posing a considerable challenge to attain.

[3] In this paper, we optimize two chillers, each with the same set of five parameters. To maintain conciseness, we select only five parameters from chiller 1 and one parameter from chiller 2 for comparison.

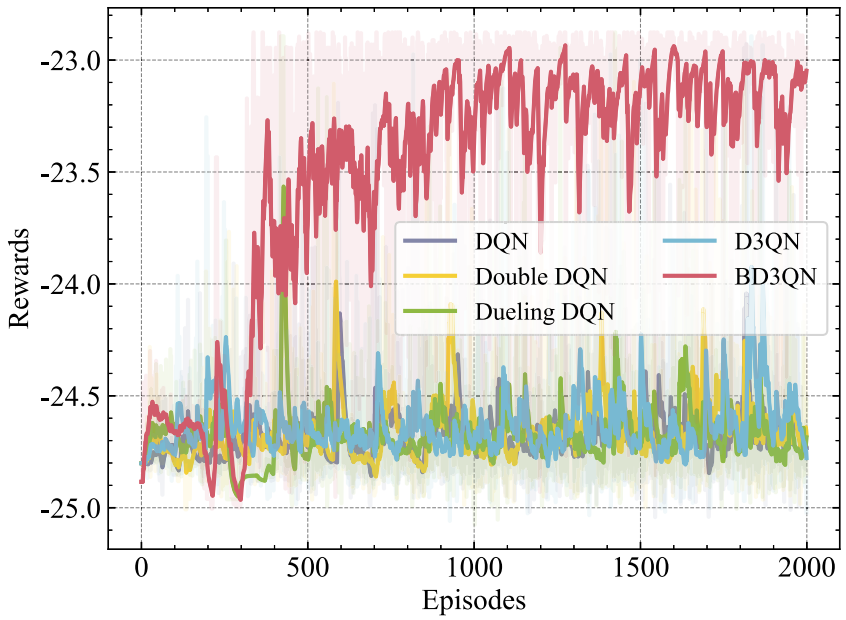

Fig. 5 Rewards variations of DRLs during the training process with 6^5 actions

governing rule for their control can be discerned. This highlights the intricacy of the system and the impracticality of human-designed strategies. Consequently, the utilization of BD3QN is deemed necessary.

7 Conclusion

In this paper, we have investigated the PUE optimization challenge within IoT-enabled data centers, employing deep learning-based algorithms. We have constructed a deep learning-powered framework tailored to PUE optimization in IoT-enabled data centers, and systematically defined a general PUE optimization problem. Subsequently, we have refined and specified this problem, focusing on the task of minimizing energy consumption in chiller cooling systems. We have developed a transformer-based prediction network to accurately predict the energy consumption of chiller cooling systems, in capable of capturing temporal dependencies. We have transformed the optimization problem into a Markov decision process and presented the branching double dueling deep Q-network to handle extremely larger action spaces while keeping good performance. Our extensive experiments have confirmed the outstanding performance of the proposed

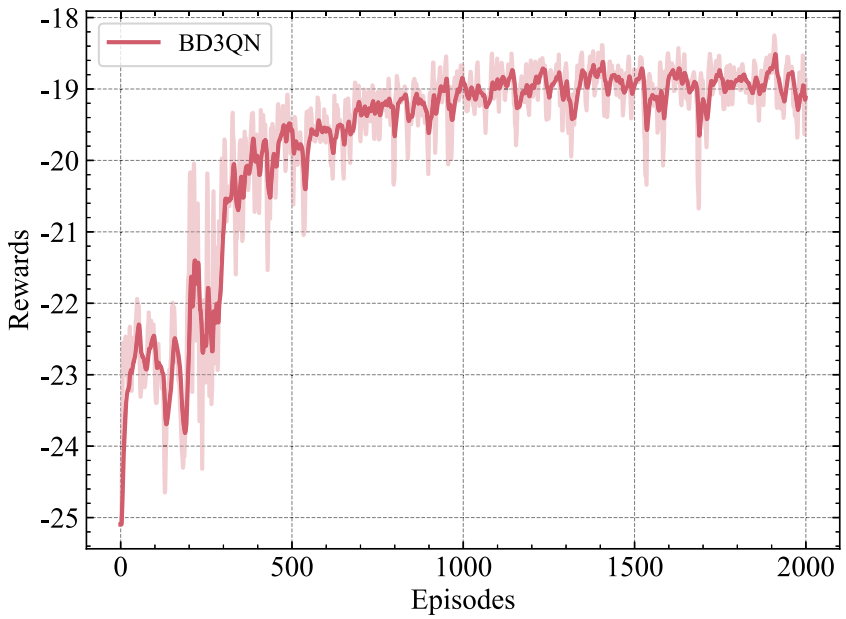

Fig. 6 Rewards variations of BD3QN during the training process with 21^{10} actions

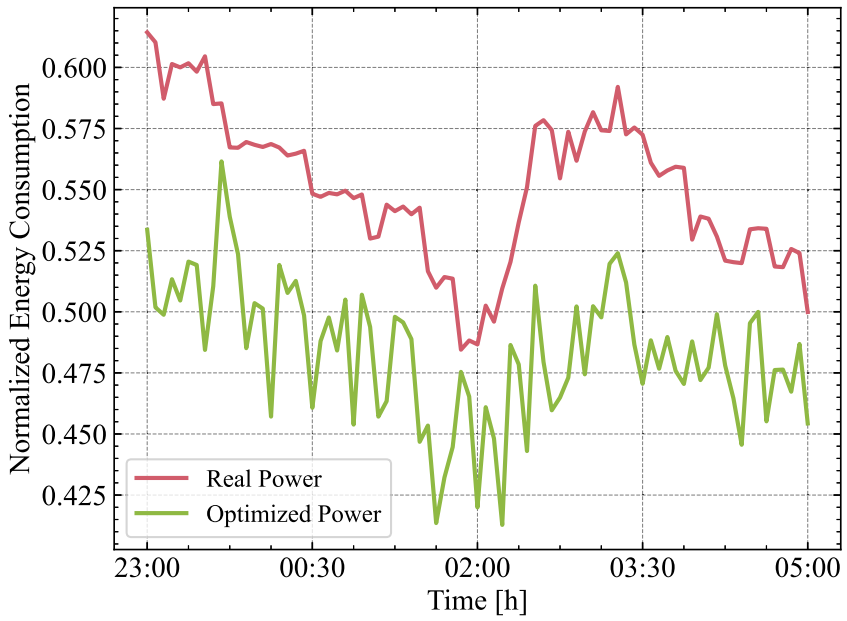

Fig. 7 Optimized energy consumption versus real energy consumption

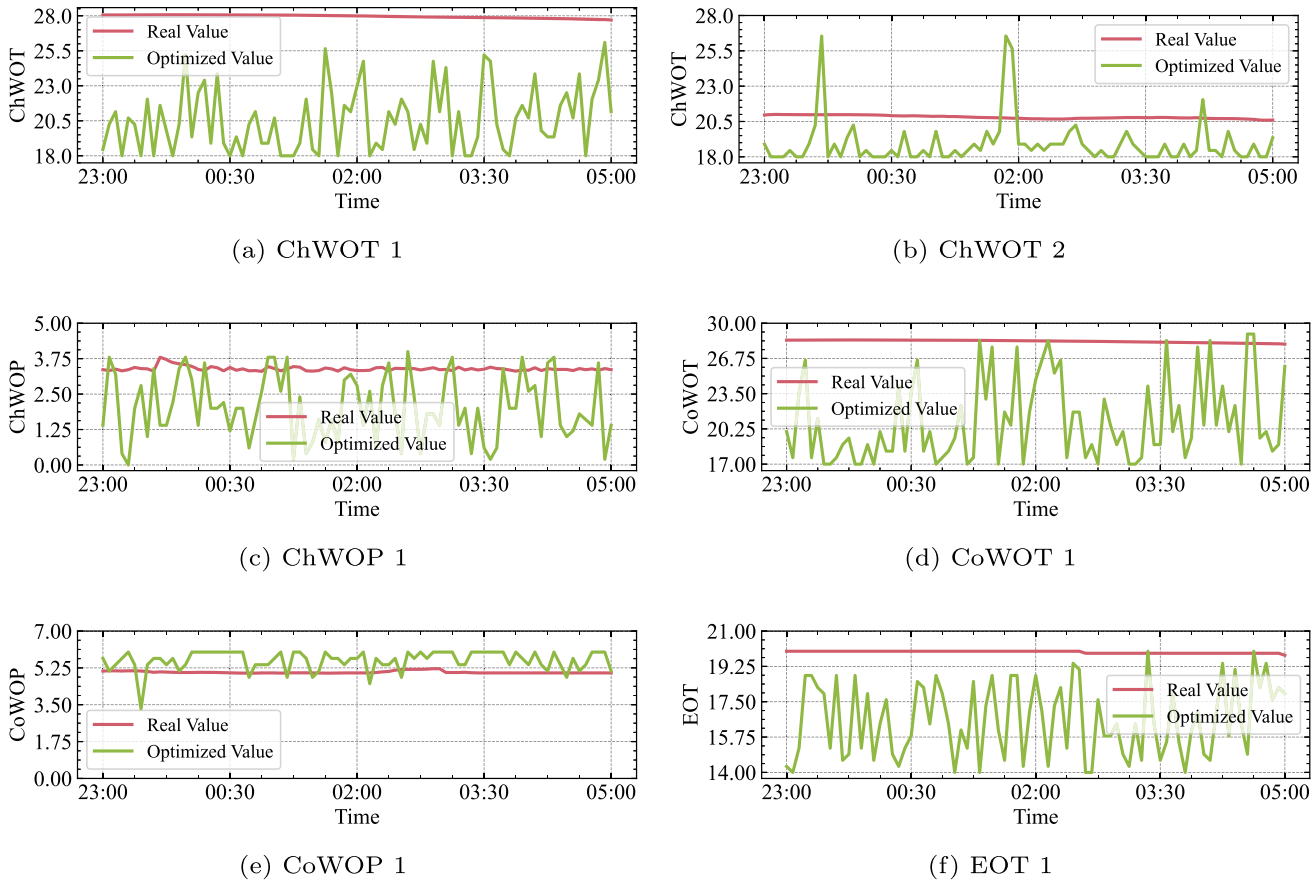

Fig. 8 Optimized parameters versus real parameters

algorithms in terms of prediction precision, optimization convergence, and optimality. In the future, we aim to further enhance our algorithms to achieve superior performance and explore additional energy-saving strategies.

Author Contributions Conceptualization, Y.S. and H.Z.; methodology, Y.S., G.J. and Y.W.; simulation, G.J. and Y.W.; formal analysis, Y.S.; investigation, H.Z.; resources, B.C. and H.Z.; data curation, G.J.; writing-original draft preparation, Y.S.; writing-review and editing, B.C., H.Z.; visualization, Y.S.; supervision, H.Z.; project administration, B.C. and H.Z.

Funding No funding was received for this study.

Availability of data and materials Not applicable.

Code availability Not applicable.

Declarations

Competing interests The authors declare no competing interests.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication I hereby declare, on behalf of my co-authors, that the work described constitutes original research which has not been previously published, in whole or in part, and is not currently under consideration for publication elsewhere. Furthermore, we affirm that informed consent from study participants has been duly obtained for the publication of the information/images in an online open-access publication.

References

1. Andrae ASG, Edler T (2015) On global electricity usage of communication technology: trends to 2030. Challenges 6(1):117–157. https://doi.org/10.3390/challe6010117
2. Cao Z, Zhou X, Hu H, Wang Z, Wen Y (2022) Toward a systematic survey for carbon neutral data centers. IEEE Communications Surveys & Tutorials 24(2):895–936. https://doi.org/10.1109/COMST.2022.3161275
3. Peter J (2021) Chinese government calls for 200 exaflops of data center compute by 2023. https://www.datacenterdynamics.com/en/news/chinese-government-calls-for-200-exaflops-of-data-center-compute-by-2023/
4. Ni J, Bai X (2017) A review of air conditioning energy performance in data centers. Renew Sustain Energy Rev 67:625–640. https://doi.org/10.1016/j.rser.2016.09.050
5. Jackson Ramphela MK, Owolawi PA, Mapayi T, Aiyetoro G (2020) Internet of Things (IoT) integrated data center infrastructure monitoring system. In: 2020 international conference on artificial intelligence, big data, computing and data communication systems (icABCD), pp 1–6. https://doi.org/10.1109/icABCD49160.2020.9183873
6. Medina-Santiago A, Azucena ADP, Gómez-Zea JM, Jesús-Magaña JA, de la Luz Valdez-Ramos M, Sosa-Silva E, Falcón-Pérez F (2020) Adaptive model IoT for monitoring in data centers. IEEE Access 8:5622–5634. https://doi.org/10.1109/ACCESS.2019.2963061
7. Xue J, Xu Y, Wu W, Zhang T, Shen Q, Zhou H, Zhuang W (2024) Sparse Mobile Crowdsensing for Cost-Effective Traffic State Estimation With Spatio-Temporal Transformer Graph Neural Network. IEEE Internet Things J 1–1. https://doi.org/10.1109/JIOT.2024.3356554
8. Wu H, Zhou H, Zhao J, Xu Y, Qian B, Shen X (2022) Deep learning enabled fine-grained path planning for connected vehicular networks. IEEE Transactions on Vehicular Technology 71(10):10303–10315. https://doi.org/10.1109/TVT.2022.3185249
9. Xiao P, Qin Z, Chen D, Zhang N, Ding Y, Deng F, Qin Z, Pang M (2023) FastNet: a lightweight convolutional neural network for tumors fast identification in mobile-computer-assisted devices. IEEE Internet Things J 10(11):9878–9891. https://doi.org/10.1109/JIOT.2023.3235651
10. Xue J, Yu K, Zhang T, Zhou H, Zhao L, Shen X (2024) Cooperative deep reinforcement learning enabled power allocation for packet duplication URLLC in multi-connectivity vehicular networks. IEEE Transactions on Mobile Computing 1–15. https://doi.org/10.1109/TMC.2023.3347580
11. Vu HD, Chai KS, Keating B, Tursynbek N, Xu B, Yang K, Yang X, Zhang Z (2017) Data driven chiller plant energy optimization with domain knowledge. In: Proceedings of the 2017 ACM on conference on information and knowledge management. CIKM '17, pp 1309–1317. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3132847.3132860
12. Yang Z, Du J, Lin Y, Du Z, Xia L, Zhao Q, Guan X (2022) Increasing the energy efficiency of a data center based on machine learning. Journal of Industrial Ecology 26(1):323–335. https://doi.org/10.1111/jiec.13155
13. Zhao P, Yang L, Kang Z, Lin J (2019) On predicting the PUE with gated recurrent unit in data centers. In: 2019 IEEE 5th international conference on computer and communications (ICCC), pp 1664–1670. https://doi.org/10.1109/ICCC47050.2019.9064040
14. Heimerson A, Sjölund J, Brännvall R, Gustafsson J, Eker J (2022) Adaptive control of data center cooling using deep reinforcement learning. In: 2022 IEEE international conference on autonomic computing and self-organizing systems companion (ACSOS-C), pp 1–6. https://doi.org/10.1109/ACSOSC56246.2022.00018
15. Li Y, Wen Y, Tao D, Guan K (2020) Transforming cooling optimization for green data center via deep reinforcement learning. IEEE Trans Cybern 50(5):2002–2013. https://doi.org/10.1109/TCYB.2019.2927410
16. Naug A, Quinones-Grueiro M, Biswas G (2022) Reinforcement learning-based HVAC supervisory control of a multi-zone building-A real case study. In: 2022 IEEE conference on control technology and applications (CCTA), pp 1172–1177. https://doi.org/10.1109/CCTA49430.2022.9966081
17. Ma Z, Wang S (2009) An optimal control strategy for complex building central chilled water systems for practical and real-time applications. Build Environ 44(6):1188–1198. https://doi.org/10.1016/j.buildenv.2008.08.011
18. Sun J, Reddy A (2005) Optimal control of building HVAC&R systems using complete simulation-based sequential quadratic programming (CSB-SQP). Building and Environment 40(5):657–669. https://doi.org/10.1016/j.buildenv.2004.08.011
19. Lu L, Cai W, Xie L, Li S, Soh YC (2005) HVAC system optimization—in-building section. Energy and Buildings 37(1):11–22. https://doi.org/10.1016/j.enbuild.2003.12.007
20. Fang Q, Wang J, Gong Q, Song M (2017) Thermal-aware energy management of an HPC data center via two-time-scale control. IEEE Transactions on Industrial Informatics 13(5):2260–2269. https://doi.org/10.1109/TII.2017.2698603
21. Balaras CA, Lelekis J, Dascalaki EG, Atsidaftis D (2017) High performance data centers and energy efficiency potential in Greece. Procedia Environ Sci 38:107–114. https://doi.org/10.1016/j.proenv.2017.03.091

22. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533. https://doi.org/10.1038/nature14236
23. Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N (2016) Dueling network architectures for deep reinforcement learning. In: Proceedings of the 33rd international conference on machine learning, pp 1995–2003. PMLR
24. Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double Q-Learning. Proceedings of the AAAI Conference on Artificial Intelligence 30(1). https://doi.org/10.1609/aaai.v30i1.10295
25. Yu K, Zhou H, Tang Z, Shen X, Hou F (2021) Deep reinforcement learning-based RAN slicing for UL/DL decoupled cellular V2X. IEEE Trans Wirel Commun 1–1. https://doi.org/10.1109/TWC.2021.3122941
26. Liu Z, Yu K, Zhang T, Liu J, Chen J, Zhou H, Shen XS (2023) Leveraging deep reinforcement learning for geolocation-based MIMO transmission in FD-RAN. In: 2023 IEEE/CIC International conference on communications in China (ICCC), pp 1–6. https://doi.org/10.1109/ICCC57788.2023.10233428
27. Ding Y, Qin X, Zhang M, Geng J, Chen D, Deng F, Song C (2023) RLSegNet: an medical image segmentation network based on reinforcement learning. IEEE/ACM Transactions on Computational Biology and Bioinformatics 20(4):2565–2576. https://doi.org/10.1109/TCBB.2022.3195705
28. Dayarathna M, Wen Y, Fan R (2016) Data center energy consumption modeling: a survey. IEEE Communications Surveys & Tutorials 18(1):732–794. https://doi.org/10.1109/COMST.2015.2481183
29. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, vol 30. Curran Associates, Inc
30. Bahdanau D, Cho K, Bengio Y (2016) Neural machine translation by jointly learning to align and translate. https://doi.org/10.48550/arXiv.1409.0473
31. Chaudhari S, Mithal V, Polatkan G, Ramanath R (2021) An attentive survey of attention models. ACM Trans Intell Syst Technol 12(5):53–15332. https://doi.org/10.1145/3465055
32. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778. https://doi.org/10.1109/CVPR.2016.90
33. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. https://doi.org/10.48550/arXiv.1607.06450
34. Tavakoli A, Pardo F, Kormushev P (2018) Action branching architectures for deep reinforcement learning. Proc AAAI Conf Artif Intell 32(1). https://doi.org/10.1609/aaai.v32i1.11798
35. Schaul T, Quan J, Antonoglou I, Silver D (2016) Prioritized experience replay. https://doi.org/10.48550/arXiv.1511.05952

Yu Sun received the B.S. degree in communication engineering from University of Electronic Science and Technology of China, Chengdu, China in 2022. He is currently pursuing the Ph.D. degree with the School of Electronic Science and Engineering, Nanjing University, Nanjing, China. His current research interests include machine learning and optimization for data center, and resource allocation for wireless communications.

Yanyi Wang received the B.S. degree in electronic information science and technology from Nanjing University, Nanjing, China in 2023. He is currently pursuing the M.S. degree with the School of Electronic Science and Engineering, Nanjing University, China. His current research interests include machine learning and optimization for data center.

Gaoxiang Jiang received the B.S. degree in communication engineering from Nanjing University, Nanjing, China in 2023. He is currently pursuing the M.S. degree with the School of Electronic Science and Engineering, Nanjing University, China. His current research interests include machine learning and optimization for data center.

Bo Cheng received the M.S. degree in signal and information processing from South China University of Technology, Guangzhou, China in 2013. He is currently pursuing the Ph.D. degree with the School of Electronic Science and Engineering, Nanjing University, Nanjing, China. His current research interests include machine learning and optimization for data center.

Haibo Zhou (M'14-SM'18) received the Ph.D. degree in information and communication engineering from Shanghai Jiao Tong University, Shanghai, China, in 2014. From 2014 to 2017, he was a Postdoctoral Fellow with the Broadband Communications Research Group, Department of Electrical and Computer Engineering, University of Waterloo. He is currently a Full Professor with the School of Electronic Science and Engineering, Nanjing University, Nanjing, China. He was elected as an IET fellow in 2022, highly cited researcher by Clarivate Analytics in 2022 & 2020. He was a recipient of the 2019 IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award, 2023–2024 IEEE ComSoc Distinguished Lecturer, 2023–2025 IEEE VTS Distinguished Lecturer, and 2023 IEEE ComSoc WTC Young Researcher Award. He served as Track/Symposium CoChair for IEEE/CIC ICCC 2019, IEEE VTC-Fall 2020, IEEE VTC-Fall 2021, WCSP 2022, IEEE GLOBECOM 2022, IEEE ICC 2024. He is currently an Associate Editor of the IEEE Transactions on Wireless Communications, IEEE Internet of Things Journal, IEEE Network Magazine, and Journal of Communications and Information Networks. His research interests include resource management and protocol design in B5G/6G networks, vehicular ad hoc networks, and space-air-ground integrated networks.