energies

MDPI

Article

Cooperatively Improving Data Center Energy Efficiency Based on Multi-Agent Deep Reinforcement Learning †

Ce Chi [1,2,‡], Kaixuan Ji [1,2,‡], Penglei Song [3], Avinab Marahatta [4], Shikui Zhang [3], Fa Zhang [1,‡], Dehui Qiu [3] and Zhiyong Liu [1,*,‡]

1 High Performance Computer Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100095, China; chice18s@ict.ac.cn (C.C.); jikaixuan@ict.ac.cn (K.J.); zhangfa@ict.ac.cn (F.Z.)
2 School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 101408, China
3 Information Engineering College, Capital Normal University, Beijing 100048, China; 2191002021@cnu.edu.cn (P.S.); 2191002065@cnu.edu.cn (S.Z.); qiudehui@cnu.edu.cn (D.Q.)
4 Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China; avinab.marahatta@iie.ac.cn
* Correspondence: zyliu@ict.ac.cn; Tel.: +86-135-2162-9531
† This paper is an extended version of our paper published in the 8th International Workshop on Energy-Efficient Data Centers (E2DC2020) at the Eleventh ACM International Conference on Further Energy Systems (e-Energy' 20), Virtual Event, Australia, 22–26 June 2020; pp. 489–495.
‡ Current address: No. 6 South Kexueyuan Rd, Beijing 100190, China.

Abstract: The problem of high power consumption in data centers is becoming more and more prominent. In order to improve the energy efficiency of data centers, cooperatively optimizing the energy of IT systems and cooling systems has become an effective way. In this paper, a model-free deep reinforcement learning (DRL)-based joint optimization method MAD3C is developed to overcome the high-dimensional state and action space problems of the data center energy optimization. A hybrid AC-DDPG cooperative multi-agent framework is devised for the improvement of the cooperation between the IT and cooling systems for further energy efficiency improvement. In the framework, a scheduling baseline comparison method is presented to enhance the stability of the framework. Meanwhile, an adaptive score is designed for the architecture in consideration of multi-dimensional resources and resource utilization improvement. Experiments show that our proposed approach can effectively reduce energy for data centers through the cooperative optimization while guaranteeing training stability and improving resource utilization.

Keywords: data center; energy efficiency; deep reinforcement learning; multi-agent; scheduling algorithm; cooling system

Citation: Chi, C.; Ji, K.; Song, P.; Marahatta, A.; Zhang, S.; Zhang, F.; Qiu, D.; Liu, Z. Cooperatively Improving Data Center Energy Efficiency Based on Multi-Agent Deep Reinforcement Learning. *Energies* **2021**, *14*, 2071. https://doi.org/ 10.3390/en14082071

Academic Editor: Marco Raugei

Received: 22 March 2021
Accepted: 6 April 2021
Published: 8 April 2021

1. Introduction

With the increasing development and application of cloud computing, data centers have become essential supporters and are located all over the world. However, the rapidly growing number and scale of data centers have created a heavy burden on the energy supply and the environment. As reported in [1], 70 billion kWh of electricity in the U.S. was consumed by data centers in 2014, which accounted for about 1.8% of all electricity used in the U.S. In addition, according to the statistics [2], data centers generate 0.3% of global carbon emissions every year. Therefore, energy consumption and environmental problems in data centers are becoming increasingly serious. To reduce the power demand and the pressure on the environment of data centers, mainly two components in data centers should be considered to be optimized, which are the IT system and the cooling system. By analysis [3], the IT system accounts for approximately 56% of the overall energy consumption in data centers, and the cooling system usually consumes about 30% of the

total energy. Therefore, for the energy consumption optimization of data centers, the focus reasonably tends to be on optimizing the IT system and the cooling system.

Mainly four challenges still exist in the energy optimization of the IT system and cooling system in data centers. Firstly, mathematical models are usually difficult to capture the details of a data center since the workload, heat and temperature in a data center change dynamically under the influence of a variety of latent factors. Therefore, the built models may inadequately describe the real environment of the data center, based on which the control algorithms in practice may not perform as well as the theoretical analysis. Secondly, when scheduling tasks for the IT system, the system state that needs to be considered could be expressed in thousands of dimensions since there could be thousands of servers in a data center. Meanwhile, the decision dimension (task scheduling) is also as high as thousands. As tasks arrive, the real-time scheduling algorithm needs to react in a very short time based on the high-dimensional state space and action space, which is an enormous challenge to optimize. Thirdly, task scheduling can be categorized as a discrete action control problem, while cooling control, such as adjusting the supply air temperature and flow rate, is a continuous action control problem. It is rather complicated to design an algorithm to simultaneously optimize both the discrete variables as well as the continuous variables. Fourthly, how to promote the collaboration between the IT system and cooling system for energy efficiency is still ambiguous. On the one hand, for the IT system, too centralized workload distribution would cause more server hot spots and increase the cooling power consumption, while too distributed workload distribution may waste more server static power. On the other hand, for the cooling system, too low supply air temperature or too high supply air flow rate will cause cooling energy waste, while too high supply air temperature or too low supply air flow rate may limit the workload distribution optimization and makes the IT system inflexible to improve the energy efficiency. Meanwhile, at every decision time, it is necessary to consider the long-term outcomes of the joint control actions, so that the energy consumption over the whole time period can be reduced, which makes the problem even harder.

In the existing researches, how to improve the energy efficiency of the IT system for data centers has been widely studied [4–7]. Various task scheduling algorithms were proposed to optimize the IT energy consumption by carefully allocating tasks to servers. Without consideration of the cooling system and the hot spot issue of servers, these methods improve the energy efficiency of the IT system but usually at the cost of increasing the cooling energy consumption. In other works, such as [8–10], adjusting the cooling facility such as the supply air temperature and flow rate to achieve the minimum cooling energy consumption is the primary concern. In these works, a slightly simplified cooling model is usually formulated to reduce the calculation overhead, based on which the cooling adjustment is optimized and the performance is verified. However, without the coordination of the IT system, the optimization capability of the cooling system is limited. Several works have been proposed to optimize the energy consumption of data centers in a joint way [11–18]. In these works, the energy consumption of both the IT system and the cooling system and even their mutual influence on each other are taken into account. Therefore, these joint optimization methods can further improve the energy efficiency of data centers. However, most of these methods still rely on abstract models of the real data center environment. Some works sacrifice performance for time, where greedy or heuristic algorithms are proposed to make scheduling decisions time-efficient.

In recent years, as deep reinforcement learning (DRL) develops fast, a novel DRL-based control method DeepEE [19] was proposed to jointly optimize the IT and cooling systems for data center energy efficiency. Based on model-free DRL, the inadequate model problem as well as the high dimensional state and action space problem can be solved elegantly. However, there still exist two problems that are not solved in DeepEE. Firstly, the joint optimization in DeepEE is achieved by optimizing the cooling action first and then calculating the scheduling action based on the cooling action. Therefore, the two-step optimization method lacks positive cooperation between the IT system and

cooling system and thus may miss the global optimal solution for the joint control problem. Secondly, DeepEE considers only CPU resource when scheduling tasks.However, resource allocation optimization can have a great influence on the energy efficiency of data centers. Training the DRL agent to optimize the multi-dimensional resources is not only a problem that has to be faced but also a conundrum. DRL is usually difficult for capturing the relationship between the server resources and task requested resources in the context of multi-dimensional resources. As a result, low resource utilization frequently occurs in the DRL-based scheduler.

In this paper, a cooperative optimization framework, multi-agent DRL-based data center cooperative control (MAD3C), is proposed to solve previous problems to further improve the energy efficiency of data centers. Specifically, DRL is adopted to train the policy by interacting with the real data center environment instead of models and to handle the high-dimensional state and action space problem. A hybrid AC-DDPG cooperative framework is proposed to jointly optimize the energy efficiency of the IT and cooling systems. Scheduling actor and critic, as well as cooling actor and critic, make up the framework, where the scheduling agents and the cooling agents are designed to cooperate with each other to optimize their policies. Thus, in the expected situation after training, the scheduling agents will tend to schedule tasks in joint consideration of the energy efficiency of the cooling system, while the cooling agents will also generate their decisions, considering the overheating protection of the servers as well as the energy efficiency of the task scheduling. Multiple dimensions of resources are considered in the DRL framework design, leading to a resource allocation optimization for the energy efficiency improvement of data centers. Training DRL for multi-dimensional resources scheduling is an extreme challenge because of the difficulty of understanding the resource relationship between servers and tasks in the DRL model. The reason according to our analysis is that the neuron network (NN) of the DRL is difficult for building the relationship between the available resources of servers and the requested resources of tasks. In our architecture, explicit information (human knowledge) is designed to provide to DRL so that the problem can be alleviated for better performance.

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

- A model-free DRL-based optimization framework MAD3C is presented to improve the energy efficiency of the IT and cooling systems in a cooperative way. Based on DRL, the strategy of MAD3C is developed via direct interaction with the real environment rather than models.
- A hybrid AC-DDPG cooperative multi-agent framework is designed to improve the cooperation between the IT and cooling systems while handling the large hybrid discrete-continuous action space. Multiple dimensions of resources are considered in the framework to improve resource utilization for higher energy efficiency.
- A scheduling baseline comparison method is devised to improve the stability of the multi-agent DRL framework and reduce computational overhead. An adaptive score scheme is designed to feed the model to improve the understanding ability of the DRL and improve resource utilization.
- We conduct experiments using the real-world traces data for the performance validation of MAD3C. Experimental results show that MAD3C is effective at saving more energy of data centers with higher resource utilization compared with state-of-the-art heuristic joint optimization algorithms and DRL-based joint control strategies.

The rest of the article is organized as follows. In Section 2, some of most related studies are introduced and analyzed. In Section 3, models and mathematical symbols of the IT system, cooling system and task are specified. The main control architecture and algorithm are presented in Section 4. A more detailed implementation design of the proposed approach is introduced in Section 5. Experimental results are presented in Section 6. Finally, Section 7 concludes the whole paper.

2. Related Work

A lot of algorithms have been proposed to improve energy efficiency for the IT system in data centers. In [4], a task scheduling algorithm is proposed for geo-distributed data centers to minimize the energy consumption and carbon emissions, considering the task deadline constraint, energy budget constraint, the maximum server number of data centers and the intermittent feature of renewable energy supply. In [5], energy-aware EnReal is proposed for scientific workflow scheduling in data centers to handle the high energy consumption problem caused by the heavy dependency and communication of scientific workflows. A balanced virtual machine (VM) scheduling algorithm is developed in [6] to realize a trade-off between energy consumption and performance when running tasks. The effect of the resource utilization prediction model is emphasized in [7], based on which a heuristic algorithm is proposed to reduce the energy consumption for data centers while guaranteeing the quality of service (QoS).

The power optimization of the cooling system has been concerned in many works as well. In [8], adaptive vent times (AVT) is considered for local cooling, and a cooling power and temperature model is built, based on which a cooling control scheme is proposed to reduce the cooling power with the constraint of rack thermal requirement. Free cooling method is considered in [9] and a cooling model containing all the main plant components is built based on First-Principle Data-Driven (FPDD) techniques. The cooling energy consumption optimization is achieved using the Particle Swarm Optimization method. To improve the cooling energy efficiency for multiple data centers, an optimized outlet temperature control strategy is developed in [10], where the high-dimensional optimization problem caused by the increasing cooler inside in-row cooling systems is handled by transforming the problem into multiple low-dimensional sub-problems and applying sequential quadratic programming (SQP) method.

More and more studies begin to consider joint energy efficiency optimization for the IT system and cooling system in data centers. In [11], a detailed problem formulation is presented, based on which an algorithm MTDP is proposed to iteratively solve integer linear programming (ILP) sub-problems to attain the optimal control of IT and cooling systems. Two major strategies, "spatial subsetting" and "inverse temperature", are trade-offs in [12], and an approach Powertrade is proposed. The servers in the data center are divided into three zones, including cool zone, warm zone and hot zone. The "spatial subsetting" strategy is applied in the cool zone servers to reduce the server idle power; the "inverse temperature" strategy is applied in the warm zone servers to eliminate the hot spots and reduce the cooling power; and the servers in the hot zone are used as candidates for the overwhelming workload. Ref. [13] considers a data center as a cyber–physical system, where a computational network is built to represent the cyber dynamics and a thermal network is built to represent the thermal dynamics in a data center. Meanwhile, three control strategies are introduced, including a baseline strategy (statically assign workloads and control the cooling system), an uncoordinated strategy (optimize the IT and cooling control separately) and a coordinated strategy (coordinately optimize the IT and cooling system and solve a non-convex problem). The three strategies were compared in the experiments and verified the effectiveness of the joint optimization of IT and cooling system. A closed-form optimal solution is analyzed in [14], where Lagrange's multiplier method with Kuhn–Tucker conditions is used to produce the optimal supply air temperature and the workload distribution. In addition, a fast online workload consolidation method is proposed to further reduce the total energy consumption of a data center. Ref. [15] considers two major problems in data center energy efficiency. The first one is maximizing the performance under a specified power consumption constraint and a thermal constraint. The second one is minimizing the power consumption while maintaining the overall performance. Both problems are solved by using a two-stage mechanism. In the first stage, the performance states (P-states) of cores, the expected execution rate of tasks and the supply air temperature of Computer Room Air Conditioners (CRACs) are optimized to satisfy the constraints of temperature, power consumption and performance. In the second

stage, optimized dynamic task scheduling is achieved to bring the actual execution rate close to the expected one. A real-time unified management approach for data center power optimization is proposed in [16]. In this work, a minimization problem considering infinite time is formulated. Lagrange relaxation and stochastic approximation methods are adopted to relieve the high calculating complexity and solve the problem. Super-linear cooling power consumption model is analyzed in [17], and a real-time task scheduling algorithm was proposed. The algorithm aims at using as few servers as possible to reduce the server power consumption and keeping load-balanced among the active servers to eliminate the hot-spots and reduce the cooling power consumption. In [18], JOINT was proposed to optimize the overall energy consumption of a data center in a cross-layer pattern. The frequency of the CPUs in the chip layer, the workload distribution as well as fan speed in the server layer and the supply air temperature in the room layer are optimized at the same time from different granularities.

However, these works still cannot solve all aforementioned challenges. As DRL has shown promising performance in many research areas, some recent studies begin to optimize task scheduling and the cooling system using DRL techniques [19–24]. In [20], a DRL-based task scheduling was proposed aiming at reducing the average job slowdown for data centers, which transformed the problem of packing tasks with multiple resource demands into a learning problem. In [21], to generate an energy-efficient job allocation policy for data centers, a training platform based on long short-term memory (LSTM) networks was built to train the DRL offline so that potential computing service quality degradation and server overheating can be avoided. The task dependency was considered in [22], and Monte Carlo Tree Search (MCTS) and DRL were used to build the scheduling framework to reduce job makespan. Based on Deep Q-Network (DQN) algorithm, a trade-off between energy consumption and service quality is achieved in [25]. LSTM is integrated into DRL to provide information about the probabilities of next states of the data center environment to generate more effective action during task scheduling. DRL was also used to optimize the cooling control policy, including fan speed and air outlet temperature to improve the cooling energy efficiency in [23,24].

A parameterized action space-based DQN algorithm DeepEE was developed in [19] to solve the hybrid action space problem and jointly optimize the job scheduling for the IT system and the airflow rate adjustment for the cooling system. However, some problems remain unsolved in DeepEE. Firstly, in DeepEE, task scheduling decisions are made after a cooling adjustment has been determined, which makes IT and cooling systems still lack sufficient cooperation, while a cooperative policy could be more energy efficient for data centers. Secondly, only CPU resource is considered in the DeepEE design. However, multi-dimensional resources task scheduling can greatly affect the data center energy efficiency since a balanced resource allocation can save more active servers and decrease the server power consumption. As the DRL is usually difficult for extracting the resource relationship between the servers and task, multi-dimensional resources scheduling makes the low resource utilization problem even worse. Based on the above observations, MAD3C is developed in this paper to improve the cooperation between IT and cooling systems and take into account multi-dimensional resources scheduling for higher energy efficiency of data centers. (Part of the work is presented at the 8th International Workshop on Energy-Efficient Data Centers (E2DC2020) at the Eleventh ACM International Conference on Further Energy Systems (e-Energy' 20), June 22-26, 2020, Virtual Event, Australia [26].)

3. System Model

In this section, the data center structure is introduced, in which models of IT system, cooling system as well as tasks are described.

3.1. Overview

As shown in Figure 1, a data center mainly consists of an IT system (servers) and a cooling system (CRAC). For the IT system, there are usually a number of users that rent

the calculating resources in data centers to run their tasks. The tasks will be firstly placed in a waiting queue in order of their arrival time and wait for their requested resources to be allocated. The task scheduler takes responsibility for choosing an appropriate server for each task, considering the QoS, energy consumption and so on. In this paper, the power consumption of the IT system is reduced by turning off idle servers. Therefore, the scheduling policy of the task scheduler is crucial for the energy efficiency of the data center. After a server is selected by the task scheduler, a virtual machine (VM) will be created to provide resources and support the running of the task. During running tasks, the servers will consume vast energy, usually supplied by the power grid. Meanwhile, the servers will generate a lot of heat which requires the cooling system to dissipate. The cooling process of the cooling system will also consume a lot of energy. As the power grid also needs to guarantee the power supply of residential electricity, industrial electricity and so on, the significant burden makes the energy reduction of data centers significant and necessary.

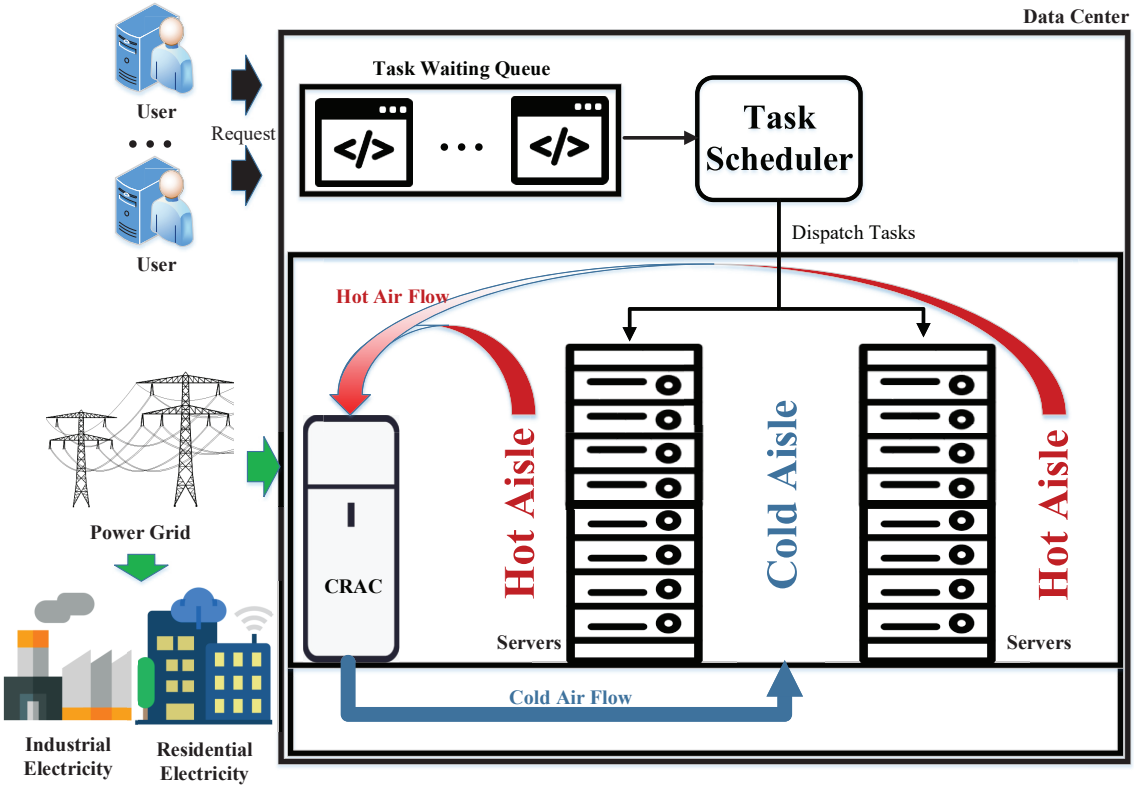


Figure 1. Structure of a data center.

The primary research goal in this paper was to reduce the energy consumption of a data center by joint IT and cooling system control. The secondary research goal includes improving the training stability of DRL and improving the resource utilization of a data center. For simplicity, we assume that a task requires exactly one VM to perform. If the execution of a task requires multiple VMs, then we can divide the task into multiple tasks with the same number of the required VMs, so that the scenario is equivalent to using one VM for a task. Therefore, the assumption will not affect the feasibility of our approach. In our design, we need temperature sensors to be installed at the outlet position of each rack to detect the server temperature. Moreover, power metering is required for both the IT system and cooling system. For the IT system, measuring the power consumption of each server is more accurate but needs a high cost of power metering equipment. Meanwhile, the server power model based on CPU utilization has been proved to be accurate with very small error [27]. Therefore, we can directly collect the CPU utilization information

of each server to deduce server power consumption for equipment expenditure savings, where the CPU utilization can be visited via various Linux commands. A brief summary of well-known power, temperature and workload monitoring equipment and software used in data centers is presented in Table 1.

Table 1. Power, temperature and workload monitoring equipment and software in data centers.

Function	Equipment	Software
Power Monitoring	Fluke 435 Power Logic PM700 APC Symmetra UPS	Dsview 3 Power Manager (Version 3.7.1, Huntsville, AL, USA)
Temperature Monitoring	SynapSense™	IBM Power Executive (Version 1.10, Armonk, NY, USA)
Workload Monitoring	top (Command) sar (Command) vmstat (Command)	HP OpenView (Version 6.01, Palo Alto, CA, USA)

3.2. IT System

Suppose that a data center is equipped with R racks. The ith rack is denoted as R_i, $i \in \{1, 2, \ldots, R\}$. Each rack R_i runs N_i servers. Let S_{ij} represent the jth server on rack R_i.

In total, D dimensions of resources of a server are considered, such as CPU, RAM, disk, etc. The total resources of a server S_{ij} can be represented as a vector $\boldsymbol{r}_{ij} = (r_{ij,1}, r_{ij,2}, \ldots, r_{ij,D})$, where $r_{ij,d}$ is the dth kind of considered resource that S_{ij} owns. When running tasks, part of the servers' resources will be allocated for tasks, so the available resources of a server S_{ij} are more useful information for task scheduling, which can be represented as $\boldsymbol{r}^a_{ij} = (r^a_{ij,1}, r^a_{ij,2}, \ldots, r^a_{ij,D})$.

Let P^s_{ij} denote the power consumption of a server S_{ij}. It can be obtained by power metering or by a generally used server power model [28] as follows for equipment expenditure savings.

$$P^s_{ij} = P^{static} + P^{dynamic} \cdot (1 - r^a_{ij,c}/r_{ij,c}), \tag{1}$$

where $r^a_{ij,c}$ and $r_{ij,c}$ represent the available CPU resource and total CPU resource of the server S_{ij}.

The outlet air temperature of each rack should be monitored to prevent servers from overheating. Suppose that the outlet air temperature of a rack R_i is measured at T^{out}_i. To protect the stability and health of the servers, a red line temperature is used to restrict the outlet air temperature of racks, which can be represented as follows.

$$T^{out}_i \leq T^{red}, \forall i \in \{1, 2, \ldots, R\}. \tag{2}$$

The power consumption of a rack R_i can be calculated by $P^r_i = \sum_{j=1}^{N_i} P^s_{ij}$. Furthermore, considering that the servers are the main power-consuming component in the IT system, the total power consumption of the IT system can be calculated as follows.

$$P^{IT} = \sum_{i=1}^{R} P^r_i = \sum_{i=1}^{R} \sum_{j=1}^{N_i} P^s_{ij}. \tag{3}$$

3.3. Cooling System

Although water cooling has been proved to be a more efficient way to cool down servers, air cooling is still a major cooling way because of its low deployment costs. Specifically, Computer Room Air Conditioner (CRAC) cooling architecture is widely used in practice and is widely studied in a lot of research [8,12,18,29,30]. Therefore, in this

paper, we consider the CRAC architecture as a case study for the cooling system energy optimization. However, we believe our designed DRL architecture can well handle different cooling scenarios. As illustrated in Figure 1, in the CRAC architecture, the racks with servers are installed on a raised floor, where perforated floor tiles are built between every two racks. When the cooling system is turned on, the cold air generated by CRACs is delivered under the raised floor to the perforated floor tiles, where cold aisles are formed. Then, the cold air comes out from the tiles and gets inhaled by the chassis fans of the servers. After that, the servers get cold down by the cold air and exhaust hot air to the hot aisles. Finally, CRACs collect the hot air and vent it out of the room.

Two factors can have a great influence on the cooling performance as well as the power consumption of the cooling system in the CRAC architecture. The first one is the supply air temperature of CRACs, denoted as T^{sup}. The other one is the supply air flow rate, denoted as f^{sup}. Lower T^{sup} and higher f^{sup} can improve the cooling effect, but will also result in more power consumption, which needs to be optimized. There are some range constraints on adjusting the two factors in practice, as shown in the following.

$$T^{sup}_{lower} \leq T^{sup} \leq T^{sup}_{upper}. \tag{4}$$

$$f^{sup}_{lower} \leq f^{sup} \leq f^{sup}_{upper}. \tag{5}$$

Suppose that after the settings of T^{sup} and f^{sup}, the cooling power consumption is monitored as P^{cool}. It is worth noting that T^{sup} and f^{sup} should be optimized not only between themselves, but also in conjunction with task scheduling, which can further reduce the total power consumption of data centers, i.e., $P^{IT} + P^{cool}$.

3.4. Task

Users run their tasks by renting resources from data centers. When a user submits a task t_k, its requested resources should be specified as a D-dimensional vector $\boldsymbol{r}^{req}_k = (r^{req}_{k,1}, \ldots, r^{req}_{k,D})$. After receiving the task, the data center should create VM on a server to provide necessary resources for the task, where the server S_{ij} should satisfy that

$$r^{a}_{ij,d} \geq r^{req}_{k,d}, \forall d \in \{1, \ldots, D\}. \tag{6}$$

For the convenience of the scheduler when selecting the server for each task, a unified ID is used to describe each server in the data center as $1, 2, \ldots, \sum_{i=1}^{R} N_i$. For a task t_k, the scheduling decision for it z can be represented as the following selection.

$$z \in \{0, 1, \ldots, \sum_{i=1}^{R} N_i\}, \tag{7}$$

where $z = 0$ indicates that the task is decided to be kept in the waiting queue this time and will be scheduled again in the next time step.

4. Joint Control

In this section, the architecture of our DRL design is provided. The DRL definitions, including state, action and reward, are made for the joint energy efficiency optimization in data centers. The details of the DRL model are described, consisting of the scheduling actor, scheduling critic, cooling actor and cooling critic. At last, the training algorithm is presented.

4.1. Overview

As illustrated in Figure 2, our designed framework includes an environment, a scheduling actor, a scheduling critic, a cooling actor and a cooling critic. The environment provides the raw state information for the agents, such as the server status, the cooling status and the requested resources of the task. Based on the state information, the scheduling actor

is responsible for calculating an appropriate decision, i.e., a server, for the current task. Similarly, the cooling actor produces an appropriate cooling adjustment, including the supply air temperature and the supply air flow rate. After the scheduling and cooling actions are performed by the environment, the next state as well as a reward is obtained and transferred to the agents. The scheduling critic and cooling critic should evaluate the decisions made by the actors based on the Markov process, which will take into account possibilities for the future and the cumulative rewards of the future. More importantly, each critic will take into account actions from both actors when evaluating, which would gradually train the actors to achieve a cooperative purpose.

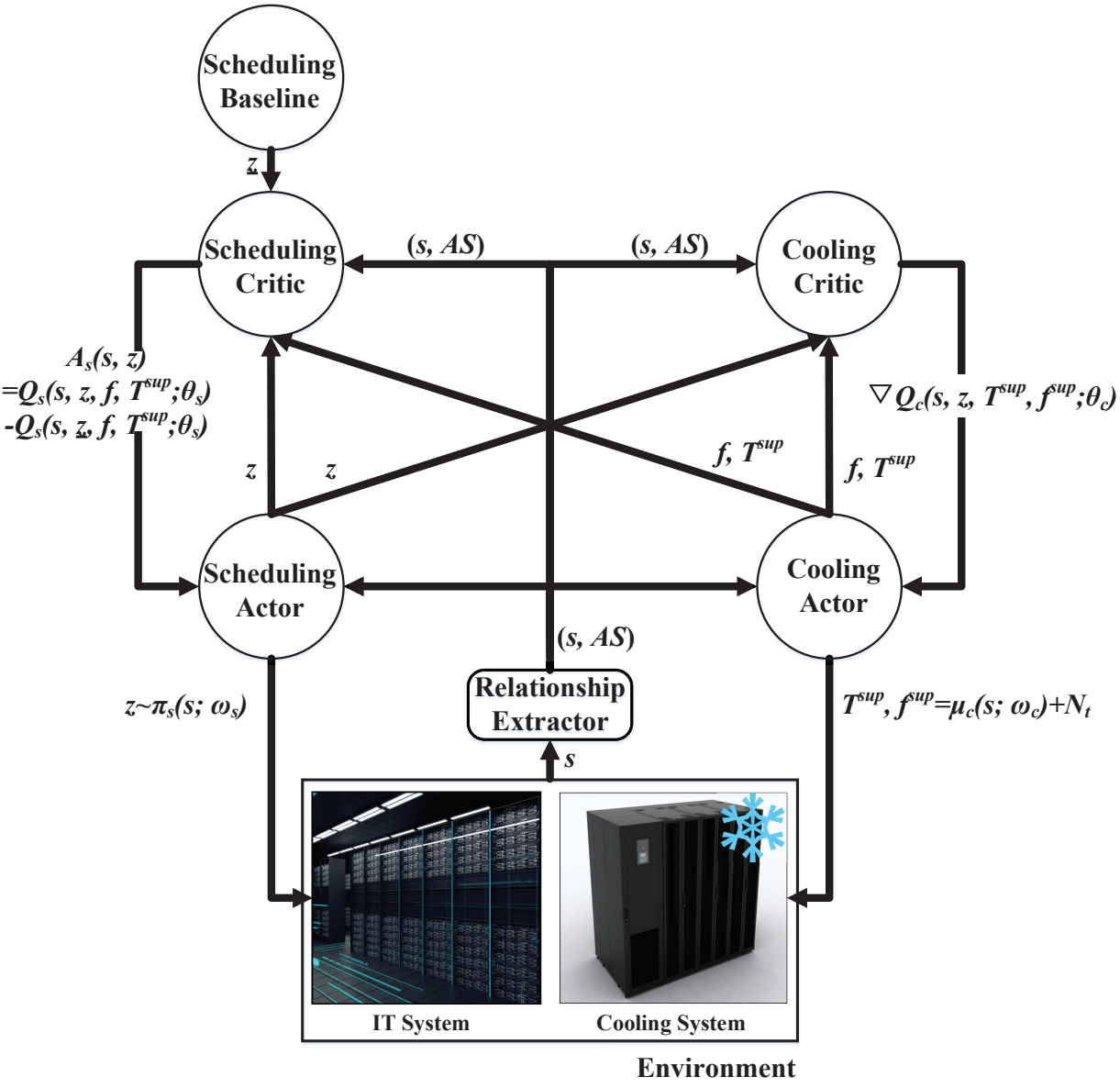


Figure 2. Structure of the deep reinforcement learning (DRL) architecture.

4.2. Definitions

4.2.1. State with Adaptability Score

Resource allocation has an essential influence on the energy efficiency of data centers. Meanwhile, multiple dimensions of resources should be taken into account for further energy efficiency improvement. However, for a DRL agent, it shows inadequate capability of extracting the relationship between the server resource and task requested resources.

As a consequence, low resource utilization often occurs in DRL-based scheduler when considering multi-dimensional resources, and an inferior energy optimization result is achieved by the agent. Fortunately, it is an easy work for us to analyze the fitness between a server and a task manually based on our experience. For instance, if the available resources of a server are larger than the task requested resources in every dimension, we can call it a "good" match. In contrast, if a dimension of the server's resources is lower than the task request resources, it is obviously a "bad" match. Human knowledge could be beneficial for the DRL agent to understand the resource relationship. Relying on the given knowledge, DRL can focus more on building the policy networks for better decisions. Therefore, an adaptability score (AS) is designed in this paper to be input into DRL as an extension of state information.

Formally, for a server, specific resources are used to compare with the requested resources of tasks, so we can define the adaptability score as follows.

$$AS(S_{ij}, t_k) = \begin{cases} 0 \text{ if } \boldsymbol{r}_{ij}^{a} < \boldsymbol{r}_{k}^{req} \\ 1 \text{ otherwise} \end{cases} \tag{8}$$

Another way to define the AS is also presented and evaluated in this paper. A soft score can be defined to represent how a server is suitable for the current task, which has shown a good performance in traditional scheduling algorithms [31]. For a specific dimension of resource d, the score is defined as

$$AS_{d}^{soft}(S_{ij}, t_k) = \begin{cases} \left(\dfrac{r_{ij,d}^{a}}{r_{k,d}^{req}} \right)^2 & \text{if } r_{ij,d}^{a} < r_{k,d}^{req} \\ 1 - \dfrac{r_{ij,d}^{a} - r_{k}^{req}}{r_{ij,d}} & \text{otherwise} \end{cases} \tag{9}$$

Based on the one-dimensional evaluation score, the eventual adaptability score of a server to the current task can be defined as the minimum score for all resource dimensions.

$$AS^{soft}(S_{ij}, t_k) = \min_{d \in \{1, \ldots, D\}} AS_{d}^{soft}(S_{ij}, t_k). \tag{10}$$

In this way, if the available resources of a server exactly match the task requested resources, AS will be 1, which means a perfect match. If the server resources are incapable of loading the task in a dimension, the AS will decrease quadratically, which denotes a not recommended match. If the server resources are more than the task requested, the AS will decrease linearly, which also limits the server available resources not to exceed the task requests too much.

In summary, the state for DRL is defined as follows.

- The available resources of each server $\boldsymbol{r}^{a} = (\boldsymbol{r}_{11}^{a}, \ldots, \boldsymbol{r}_{RN_R}^{a})$.
- The power consumption of each server $\boldsymbol{P}^{s} = (P_{11}^{s}, \ldots, P_{RN_R}^{s})$.
- The adaptability scores of each server to the current task $\boldsymbol{AS} = (AS_{11}, \ldots, AS_{RN_R})$.
- The outlet air temperature of each rack $\boldsymbol{T}^{out} = (T_{1}^{out}, \ldots, T_{R}^{out})$.
- The power consumption of each rack $\boldsymbol{P}^{r} = (P_{1}^{r}, \ldots, P_{R}^{r})$.
- The supply air temperature and flow rate T^{sup} and f^{sup}.
- The requested resources of the current task $\boldsymbol{r}_{k}^{req} = \{r_{1}^{req}, \ldots, r_{D}^{req}\}$.

Finally, the state is defined as follows.

$$\boldsymbol{s} = (\boldsymbol{r}^{a}, \boldsymbol{P}^{s}, \boldsymbol{AS}, \boldsymbol{T}^{in}, \boldsymbol{P}^{r}, T^{sup}, f^{sup}, \boldsymbol{r}_{k}^{req}). \tag{11}$$

4.2.2. Power-Based Reward Design

The reward design is greatly critical to the DRL, because not only the targets but also the training stability of the DRL should be considered in the reward design. DeepEE [19] adopted Power Usage Efficiency (PUE) as the major criteria and added PUE to the reward design. However, using PUE to direct the DRL may cause additional energy waste. This is because, due to the definition of PUE (PUE $\approx (P^{IT} + P^{cool})/P^{IT}$), decreasing the PUE not always indicates the reduction of the total energy consumption of the data center. For instance, increasing P^{IT} can also decrease the PUE value. Based on this observation, in this paper, the main target of the DRL is set to minimize the total energy consumption including both the IT system and cooling system. However, directly adopting the raw value of the total power consumption may lead to an unstable performance. The main reason is that with the end of some tasks, the power consumption will naturally decrease, which is not caused by the DRL's actions. As a consequence, a nasty action made by the DRL may still obtain a high score with fewer tasks in the servers, misleading the training of the DRL. Therefore, a differential value of the power consumption is used in our model design to train the DRL, which represents the direct power influence caused by the actions. In this way, the training of DRL is stabilized. For the remaining parts of the reward function, the quality of service (QoS) should be guaranteed as well. In this paper, the task waiting time and the server resources are considered to optimize to guarantee the QoS and are added into the reward design. Finally, to protect the stability and health of servers, the outlet air temperature of servers should also be constrained. In summary, the reward function is designed in the following way.

$$\begin{aligned} Reward =& \beta - \beta_1(\Delta P^{IT} + \Delta P^{cool}) \\ & -\beta_2 w_k \\ & -\beta_3 \sum_{i=1}^{R} \max\{0, T_i^{out} - T^{red}\}/R \\ & -\beta_4 \min\{\alpha^{limit}, \sum_{d=1}^{D} (\exp(r_d^{QoS} - r_{z,d}^a) - 1)\}, \end{aligned} \tag{12}$$

where β_1, β_2, β_3 and β_4 denote the weight of each target and allow the data center operator to make trade-offs among the targets. ΔP^{IT} and ΔP^{cool} are difference values of P^{IT} and P^{cool} after and before the action performed. w_k is the time that the current task has been waiting for, i.e., the current time minus the arrival time of the task. Note that the existence of w_k in the reward design is necessary for DRL, because a DRL will gradually learn a policy that does not schedule any task without w_k. $r_{z,d}^a$ is the available resource that the target server holds. r_d^{QoS} is a constant, used to avoid resource overload of servers and guarantee the QoS. α^{limit} is used to limit the overloading penalty to not being too large. β equalizes the distribution of positive and negative rewards.

4.2.3. Hybrid Action Control

The scheduling decision is defined as $z \in \{0, 1, \ldots, \sum_{i=1}^{R} N_i\}$. $z \in \{1, \ldots, \sum_{i=1}^{R} N_i\}$ means a server ID that the task will be scheduled to. $z = 0$ means that the task will be moved to the back of the waiting queue and will be rescheduled at the next scheduling time step. The cooling decision consists of the supply air temperature T^{sup} and the flow rate f^{sup} under the constraints (4) and (5). In the end, the actions z, T^{sup} and f^{sup} should be optimized jointly to improve the overall energy efficiency of the data center. Classic DRL algorithms are usually incapable of the hybrid discrete-continuous actions optimization. Although in DeepEE, a PADQN algorithm is proposed to solve this problem, it is achieved by producing the continuous action (cooling adjustment) first, and calculating the discrete action (scheduling) based on the continuous action. The two-step optimization method may cause a lack of cooperation between the cooling and IT systems and miss the global

optimum. Therefore, a novel DRL architecture is designed in this paper to handle these problems, which is presented in the following subsection.

4.3. DRL Model

Classic and widely-used DRL algorithms include DQN [32], Actor-Critic (AC) [33] and Deep Deterministic Policy Gradient (DDPG) [34]. DQN is suitable for discrete action space problems, while DDPG is designed for continuous action space problems. AC is capable of both discrete and continuous action space problems, but is not as stable as DQN and DDPG. In conclusion, these DRL algorithms can well handle either a discrete action space problem or a continuous action space problem, but are unable to solve a hybrid discrete-continuous action space problem like joint energy optimization of data centers. Meanwhile, multi-agent cooperative DRL algorithms such as [35] tend to be more and more stable and sophisticated, where good cooperation among the agents is achieved for a specific mission. However, these algorithms are usually built based on homogeneous agents so that pure discrete actions or continuous actions are optimized, still not supporting a discrete-continuous hybrid actions problem. Therefore, our solution is to build a multi-agent cooperative architecture where some agents take responsibility of the discrete action while other agents produce continuous action. How to maintain the stability and performance of such a hybrid architecture becomes an extreme difficulty, and we propose several methods to solve the problems.

To be specific, in our design, AC structure is adopted to optimize the discrete scheduling decision z. As for the cooling adjustment problem, DDPG structure is leveraged to produce an optimal decision of the set-points of supply air temperature T^{sup} and f^{sup}, which are continuous actions. DQN is abandoned in our design since it cannot well fit into our multi-agent framework. To improve the cooperation between task scheduling agents and cooling agents, an AC-DDPG cooperative multi-agent architecture is developed in this paper, so that the decisions of task scheduling and cooling adjustment can be jointly optimized and higher energy efficiency could be achieved for data centers.

On the whole, as shown in Figure 3, a scheduling actor, a scheduling critic, a cooling actor and a cooling critic are constructed and interact with the data center environment constantly to train their policies. The scheduling actor takes the state as input and outputs the scheduling action z. Similarly, the cooling actor inputs the state and outputs the cooling action T^{sup} and f^{sup}. All of the state, the scheduling action and the cooling action are inputted into the scheduling critic to evaluate how the actions are under the specified state. To be concrete, only when both the scheduling and the cooling actions are conducive to the cumulative reward will the evaluated score be high. Then the score will be used to train the scheduling actor. In addition, the cooling critic is responsible to produce the score for the training of the cooling actor, and also takes into account both the scheduling action and cooling action with the state. In addition, a scheduling baseline comparison method is developed to stabilize the structure. Detailed description for each part is provided in the following subsections.

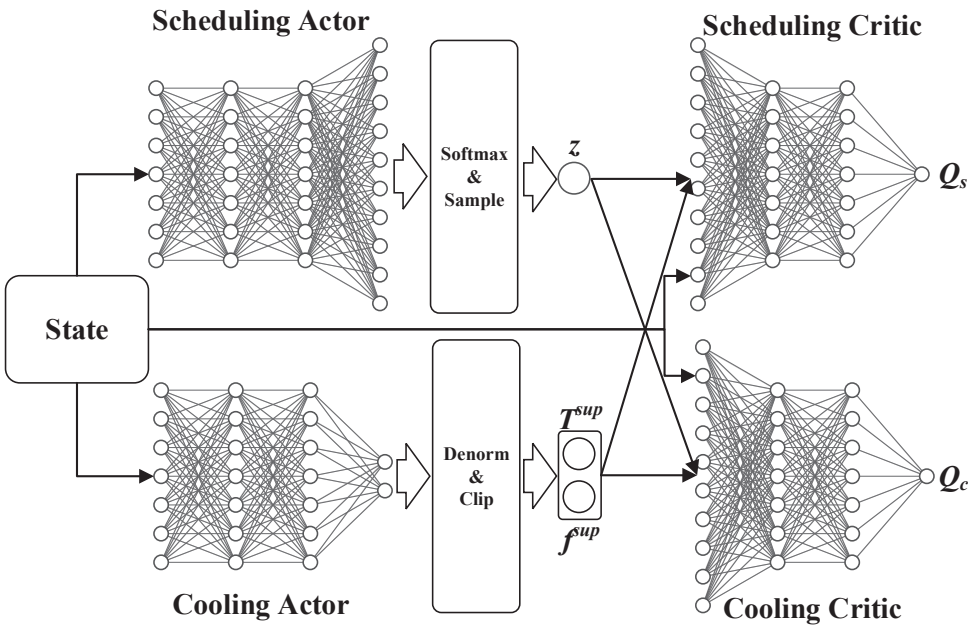


Figure 3. The neural networks of the multi-agent DRL architecture.

4.3.1. Scheduling Actor

The scheduling actor is constructed by taking the actor model in the AC algorithm as the prototype. After receiving the state vector $\boldsymbol{s}$, the scheduling actor calculates the corresponding action by feeding the state $\boldsymbol{s}$ into its neural network (NN) with parameters of ω_s: $z \sim \pi_s(\boldsymbol{s}, z; \omega_s)$. Specifically, the output of the NN is represented as a vector with the dimension of $\sum_{i=1}^{R} N_i + 1$. Each element in the vector is a probability in the range of $[0, 1]$ and the sum of the elements is 1 which is achieved through the Softmax process. The scheduling action is generated by sampling the vector, and finally an integer z is obtained. Based on z, if $z = 0$, the task will be moved to the back of the waiting queue and will be rescheduled at the next scheduling time step; and if $z \in \{1, \ldots, \sum_{i=1}^{R} N_i\}$, it represents the ID of the server that the task will be assigned to.

4.3.2. Cooling Actor

The cooling actor is developed on the basis of the actor structure in the DDPG algorithm. To produce the cooling decision including T^{sup} and f^{sup}, the state s is input to the evaluation network with parameters of ω_c. Then, T^{sup} and f^{sup} are obtained by combining the NN outputs $\mu_c(\boldsymbol{s}; \omega_c)$ with a random value $\mathcal{N}_t$ which is added for exploration of DRL. Finally, the process can be formulated as $(T^{sup}, f^{sup}) = \mu_c(\boldsymbol{s}; \omega_c) + \mathcal{N}_t$. In addition, a target network of the cooling actor $\hat{\omega_c}$ is implemented and get updated with the training of the cooling actor, which is essential for the stability of the DDPG training.

4.3.3. Scheduling Critic with Scheduling Baseline Comparison Method

The backbone of the scheduling critic is based on the critic structure of the AC algorithm. Differently, in order to achieve cooperation between the task scheduling and cooling adjustment, the scheduling critic evaluates not only the action produced by the scheduling actor z, but also the actions that come from the cooling actor (T^{sup}, f^{sup}). When making an evaluation, the state $\boldsymbol{s}$ as well as z and (T^{sup}, f^{sup}) are input into the NN of scheduling critic with parameters of θ_s. After that, a score that evaluates the actions z and (T^{sup}, f^{sup}) in the state s is produced, i.e., $Q_s(\boldsymbol{s}, z, T^{sup}, f^{sup}; \theta_s)$.

A significant problem that needs to be solved is that the evaluated score is for both the task scheduling action z and the cooling action (T^{sup}, f^{sup}), which cannot be used directly to train the scheduling actor. Otherwise, instability and inefficiency would occur. For instance, the scheduling actor may provide a good scheduling action while the cooling actor provides a bad one. For the scheduling critic, the overall score for both actions will not be high because the combination of the actions are not good for the joint control. If the score is used to train the scheduling actor, it will mislead the scheduling actor and decrease the possibility of making a good action. As a consequence, not only would the training be unstable, but also the performance of DRL would decrease.

Although existing research [36] has explored a counterfactual baseline method to solve this problem, it becomes inefficient in the data center joint energy optimization problem. The reason is that the counterfactual baseline method requires all the possible actions to be passed through the scheduling critic network, which will cause tremendous overheads since there could be thousands of servers (actions) in a data center. In order to provide an appropriate evaluation of the scheduling action for the training of the scheduling actor in an efficient way, a scheduling baseline comparison method is proposed in this paper: Firstly, an existing task scheduling algorithm is specified as the baseline algorithm, which can be a heuristic algorithm like scheduling tasks to a server in the coolest rack (i.e., Coolest Server First or CSF for short) or selecting a server in a Round Robin (RR) routine. Secondly, in each scheduling decision time step, the baseline algorithm produces its decision of task scheduling $\underline{z}$ based on the current state $\boldsymbol{s}$. Thirdly, the action produced by the baseline algorithm is fed to the scheduling critic and output the corresponding score $Q_s(\boldsymbol{s}, \underline{z}, T^{sup}, f^{sup}; \theta_s)$. Finally, an advantage score is calculated as follows.

$$A_s(\boldsymbol{s}, z) = Q_s(\boldsymbol{s}, z, T^{sup}, f^{sup}; \theta_s) - Q_s(\boldsymbol{s}, \underline{z}, T^{sup}, f^{sup}; \theta_s). \tag{13}$$

Afterwards, $A_s(\boldsymbol{s}, z)$ is used to train the scheduling actor. Based on the theory of AC algorithm, the update gradient of the scheduling actor can be determined by

$$\nabla_{\omega_s} J_s = A_s(\boldsymbol{s}, z) \cdot \nabla_{\omega_s} \log \pi_s(\boldsymbol{s}, z; \omega_s). \tag{14}$$

In this way, the calculation complexity of the scheduling actor is reduced from $O((\sum_{i=1}^{R} N_i + 1) \cdot F)$ to $O(2F)$ if the CSF or RR scheduling method is adopted as the baseline algorithm. F represents the Floating-point Operations (FLOPs) of the NN, which depend on the NN design.

Similarly, based on the theory of AC algorithm, the NN update of the scheduling critic is calculated by minimizing the mean square loss as follows.

$$L_s(\theta_s) = (Q_s(\boldsymbol{s}, z, T^{sup}, f^{sup}; \theta_s) - y_s)^2, \tag{15}$$

where

$$y_s = r + \gamma \cdot Q_s(\boldsymbol{s}', z', f', T^{sup\prime}; \theta_s), \tag{16}$$

and $z' \sim \pi_1(\boldsymbol{s}', z'; \omega_1)$ and $(f', T^{sup\prime}) = \mu_c(\boldsymbol{s}'; \hat{\omega_c})$. γ is the discount factor which is widely used in Markov Decision Process (MDP) models and DRL algorithms.

4.3.4. Cooling Critic

DDPG critic is leveraged in our cooling critic design. The cooling critic is set to evaluate the cooling actions (T^{sup}, f^{sup}) in combination with the action of the scheduling actor z for cooperation. It inputs the state s, scheduling action z and cooling actions (T^{sup}, f^{sup}) into its evaluation networks with parameters of θ_c. After that, an evaluation score is produced $Q_c(\boldsymbol{s}, z, T^{sup}, f^{sup}; \theta_c)$, which will be used for the training of the cooling critic as well as the cooling actor.

As for training, firstly, B transitions are sampled from the replay buffer, denoted as $\mathcal{B} = \{(\boldsymbol{s}_j, z_j, T_j^{sup}, f_j^{sup}, r_j, \boldsymbol{s}'_j)\}_{1 \le j \le B}$. Then the evaluation network of the cooling critic is updated by applying gradient descent method to minimize the mean square error:

$$L_c(\theta_c) = \frac{1}{B} \sum_{j=1}^{B} (Q_c(\boldsymbol{s}_j, z_j, T_j^{sup}, f_j^{sup}; \theta_c) - y_{c,j})^2, \tag{17}$$

where $y_{c,j}$ is the target Q-value, calculated by the cooling critic target network $\hat{\theta}_c$, which can be formulated as follows.

$$y_{2,j} = r_j + \gamma \cdot Q_c(\boldsymbol{s}'_j, z'_j, T_j^{sup\prime}, f^{sup\prime}; \hat{\theta}_c), \tag{18}$$

where $z'_j \sim \pi_s(\boldsymbol{s}'_j, z'_j; \omega_s)$ and $(T_j^{sup\prime}, f^{sup\prime}) = \mu_c(\boldsymbol{s}'_j; \hat{\omega}_c)$.

After that, based on the DDPG framework, the update gradients for ω_c of the cooling actor can be calculated by

$$\begin{aligned} \nabla_{\omega_c} J_c = \frac{1}{B} \sum_{j=1}^{B} & \nabla_{(T^{sup}, f^{sup})} Q_c(\boldsymbol{s}_j, z_j, T^{sup}, f^{sup}; \theta_c)|_{(T^{sup}, f^{sup}) = \mu_c(s_j; \omega_c)} \\ & \cdot \nabla_{\omega_c} \mu_c(\boldsymbol{s}_j; \omega_c) \end{aligned} \tag{19}$$

At last, the parameters of the cooling actor and critic target networks $\hat{\omega}_c$ and $\hat{\theta}_c$ are updated using soft update method with parameter τ [34].

4.4. Algorithm

Finally, the algorithm is presented in Algorithm 1. From lines 6 to 9, the scheduling action and cooling action are obtained from the actors, performed by the environment, and stored in a list with other information. The scheduling critic is trained at lines 11–12. After that, a mini-batch is sampled and the cooling critic is trained at lines 13–15. At line 16, the scheduling actor is updated based on the scheduling critic and the baseline action obtained at line 10. Moreover, the cooling actor is updated according to Equation (19) at line 17. Finally, the cooling actor and critic target networks are updated in a soft update method at lines 18. In addition, the whole flowchart of the algorithm is illustrated in Figure 4.

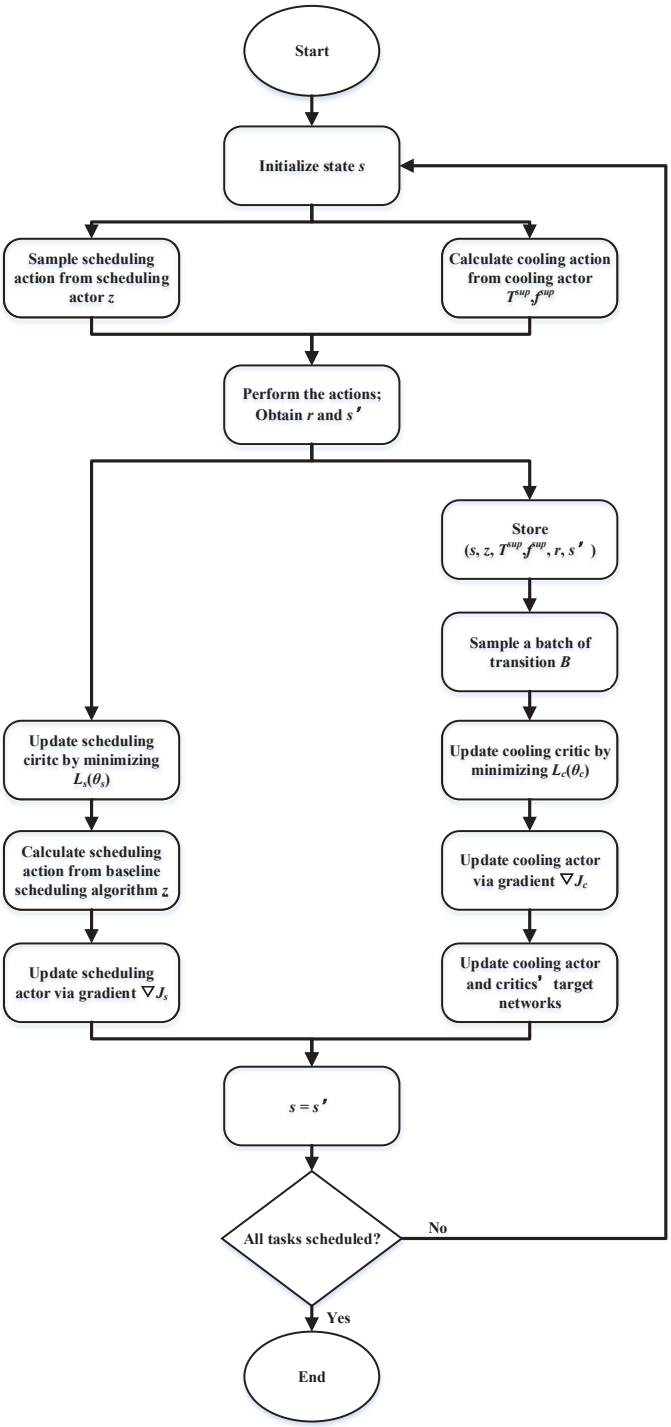


Figure 4. Flow diagram of MAD3C.

Algorithm 1 MAD3C

Initialize the NNs of scheduling actor $\pi_s(\boldsymbol{s}, z; \omega_s)$, scheduling critic $Q_s(\boldsymbol{s}, z, T^{sup}, f^{sup}; \theta_s)$, cooling actor $\mu_c(\boldsymbol{s}; \omega_c)$ and cooling critic $Q_c(\boldsymbol{s}, z, T^{sup}, f^{sup}; \theta_c)$ and their parameters ω_s, θ_s, ω_c and θ_c.
Initialize the cooling actor target networks $\mu_c(\boldsymbol{s}; \hat{\omega_c})$ and cooling critic target networks $Q_c(\boldsymbol{s}, z, T^{sup}, f^{sup}; \hat{\theta_c})$ and their parameters $\hat{\omega_c} = \omega_c$ and $\hat{\theta_c} = \theta_c$.
$\mathcal{R} = \{\}$.
$\boldsymbol{s}$= initial state.
for $t = 1$ to T **do**
  $z \sim \pi_1(\boldsymbol{s}, z; \omega_1)$
  $(T^{sup}, f^{sup}) = \mu_c(\boldsymbol{s}; \omega_c) + \mathcal{N}_t$
  The environment executes actions z, T^{sup} and f^{sup}, and calculates the reward r and observes next state $\boldsymbol{s}'$.
  Save the current transition $(\boldsymbol{s}, z, T^{sup}, f^{sup}, r, \boldsymbol{s}')$ into $\mathcal{R}$.
  The baseline scheduling algorithm produces an action $\underline{z}$ based on the state s.
  $y_s = r + \gamma \cdot Q_s(\boldsymbol{s}', z', T^{sup\prime}, f^{sup\prime}; \theta_s)|_{z' \sim \pi_s(\boldsymbol{s}', z'; \omega_s), (T^{sup\prime}, f^{sup\prime}) = \mu_c(\boldsymbol{s}'; \hat{\omega_c})}$
  Train the scheduling critic by minimizing $L_s(\theta_s) = (Q_s(\boldsymbol{s}, z, T^{sup}, f^{sup}; \theta_s) - y_s)^2$.
  Randomly select a minibatch of historical transitions $\mathcal{B} = \{(\boldsymbol{s}_j, z_j, T_j^{sup}, f_j^{sup}, r_j, \boldsymbol{s}_j')\}_{1 \le j \le B}$ from $\mathcal{R}$.
  $y_{c,j} = r_j + \gamma \cdot Q_c(\boldsymbol{s}_j', z_j', T_j^{sup\prime}, f_j^{sup\prime}; \hat{\theta_c})|_{z_j' \sim \pi_s(\boldsymbol{s}_j', z_j'; \omega_s), (T_j^{sup\prime}, f_j^{sup\prime}) = \mu_c(\boldsymbol{s}_j'; \hat{\omega_c})} \quad j = 1, \ldots, B$
  Train the cooling critic by minimizing $L_c(\theta_c) = \frac{1}{B} \sum_{j=1}^{B} (Q_c(\boldsymbol{s}_j, z_j, T_j^{sup}, f_j^{sup}; \theta_c) - y_{c,j})^2$.
  Update the scheduling actor ω_s by applying the update gradient: $\nabla_{\omega_s} J_s = (Q_s(\boldsymbol{s}, z, T^{sup}, f^{sup}; \theta_s) - Q_s(\boldsymbol{s}, \underline{z}, T^{sup}, f^{sup}; \theta_s)) \cdot \nabla_{\omega_s} \log \pi_s(\boldsymbol{s}, z; \omega_s)$.
  Update cooling actor ω_c using the gradient: $\nabla_{\omega_c} J_c = \frac{1}{B} \sum_{j=1}^{B} \nabla_{(T^{sup}, f^{sup})} Q_c(\boldsymbol{s}_j, z_j, T^{sup}, f^{sup}; \theta_c)|_{(T^{sup}, f^{sup}) = \mu_c(s_j; \omega_c)} \cdot \nabla_{\omega_c} \mu_c(\boldsymbol{s}_j; \omega_c)$.
  Update the cooling actor critic target networks:
  $\hat{\omega_c} = \tau \omega_c + (1 - \tau) \hat{\omega_c}$
  $\hat{\theta_c} = \tau \theta_c + (1 - \tau) \hat{\theta_c}$
  $\boldsymbol{s} = \boldsymbol{s}'$
end for

5. Implementation

In this section, a more detailed system implementation design of MAD3C is presented. As illustrated in Figure 5, a task waiting queue, a data preparation module, a historical transition database (HTDB), a state information database (SIDB), a server interface, a cooling interface, a trainer, a scheduling actor, a scheduling critic, a cooling actor and a cooling critic make up the system. The server monitor collects the information of servers including the remaining available resources $\boldsymbol{r}^a$ and the power consumption of each server $\boldsymbol{P}^s$. The cooling monitor collects the information about the outlet air temperature of each rack $\boldsymbol{T}^{in}$, the power consumption of the cooling system P^{cool} and the supply air temperature T^{sup} with flow rate f^{sup}. After that, the information is written into SIDB, waiting to be acquired. When a user submits a task to the data center, the task is placed into a task waiting queue in the descending order of the task arrival time. Initially, the data preparation module gets information of a task $\boldsymbol{r}_k^{req}$ from the task waiting queue every time. Then, part of the state information is obtained from the SIDB and the remaining state is calculated and supplemented by the data preparation module, such as the power consumption of each rack $\boldsymbol{P}^r$ and $\boldsymbol{AS}$. After normalizing the state, the data preparation module feeds the state vector $\boldsymbol{s}$ to the scheduling actor and the cooling actor. The scheduling actor and cooling actor calculate their actions z and (T^{sup}, f^{sup}) through their NNs as described in Section 4.3. The actions are then transferred to the scheduling executor and cooling executor to perform the corresponding task scheduling and the cooling adjustment. To make the system able to

be deployed in practice, an overloading protector module is added into the system whose responsibility is to check whether the target server has enough resources to accept the task. If overloading would occur after the allocation, the scheduling action will be rejected, and the task will be scheduled in a Round Robin (RR) routine. Similarly, an overheating protector module is added to monitor the system heat state. A model-based heat prediction algorithm is embedded into the overheating protector module. If the following cooling adjustment would cause overheating according to the prediction, the adjustment will be rejected and a lower T^{sup} with a higher f^{sup} will be set to prevent latent overheating. After the scheduling and cooling adjustment, the SIDB is refreshed. Next, the data preparation module fetches the state information, calculates the reward of the performed actions and stores the previous state, actions, reward and the current state into HTDB. For training the agents, a trainer module gets a batch of historical data from HTDB every time and updates the NNs of the scheduling critic, cooling critic, scheduling actor and cooling actor. Note that when purely inferring the actions, the critics are not needed, which can further reduce the amount of calculation and increase the efficiency of our architecture.

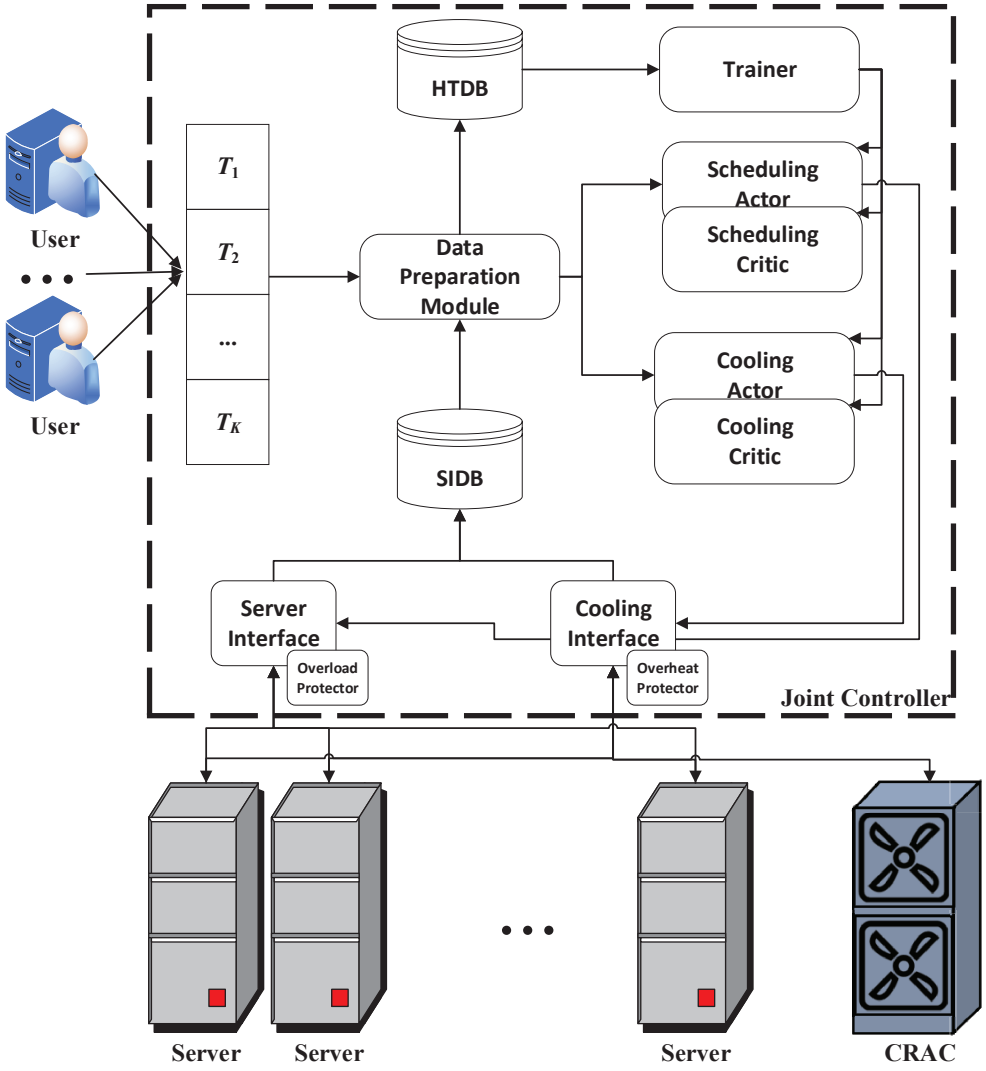


Figure 5. The DRL-based joint control system design.

6. Experiments

In this section, experiments are performed to validate the effectiveness of MAD3C. The experiment settings are given first. Then, experimental results are presented.

6.1. Settings

We construct a simulation environment based on the OpenAI gym template. The DRL algorithm is built via Python and Tensorflow. The simulation is conducted on a server equipped with an Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10 GHz CPU, 125 GB of RAM and 2 Tesla K20c GPUs. A data center with 20 racks and 840 servers is considered for the evaluation. The static power and dynamic power of each server are set to 100 W and 200 W [19], respectively. Since CPU and RAM resources are the most in short supply in data centers [31], we mainly take into account CPU and RAM for scheduling. The cooling environment is built based on the CRAC cooling model, which is introduced in [37], where the cooling matrix is generalized based on norm random distribution with mean of 0.1 and variance of 0.05. This model is designed as an alternative to traditional Computational Fluids Dynamics (CFD) with a small error to enable fast computation to support online temperature prediction and task scheduling decisions. It takes rack power consumption as input and produces the outlet air temperature for each rack so that the temperature signals can be utilized by DRL agents to train to avoid overheating. We implement the model via the Python Numpy library to accelerate the matrix calculation. The red line temperature is set to 30 °C [19]. The task trace is extracted from widely-used Google cluster data [38], where 10,000 tasks are used to train the DRL neuron networks, 8000 tasks for validation and 8000 tasks for test. 3-layer fully-connected NN is used to build scheduling actor, scheduling critic, cooling actor and cooling critic. More detailed parameters are given in Table 2. For the scheduling critic, the CSF routine is used as the task scheduling baseline method for MAD3C. The influence of different baseline algorithms for MAD3C is also illustrated and analyzed in the experiment results part. In comparison with other algorithms, we mainly adopt the integer model of AS (Equation (8)) first. More results about the influence of different AS models are provided and analyzed in the experiments as well. During the simulation, critical information such as the reward for each action, the loss values of the agents, the IT and cooling power at each time step, the resource utilization of each server, the waiting time of each task, the supply and outlet air temperature of each rack and so on is recorded into files. After collection, the data are collated and plotted by Matlab.

6.2. Baselines

Mainly 4 different task scheduling and cooling control schemes are used to compare for effectiveness validation of our method. (1) Random: A random scheduling policy with a random cooling control policy. It is evaluated in the experiments to compare with our methods and validate that the DRL agents can form an effective control policy. (2) RR: a traditional scheduling scheme that selects servers in sequence for tasks. The cooling adjustment is based on a simple model that would make the outlet temperatures of the racks just below the upper-temperature limit by adjusting the supply air temperature and flow rate based on its prediction, where inaccuracy may exist since the model may be inadequate to describe the complicated data center environment. (3) PowerTrade: A joint optimization approach that spreads tasks out in places prone to overheating to reduce hot spots, and centralizes tasks in cooler places to save server idle power. It is implemented in our CRAC-based experiments according to its core idea. (4) DeepEE: A joint control approach utilizing DRL for action optimization, which decides the cooling action first and then calculates the scheduling action based on the cooling action. This strategy still lacks cooperation and may miss the global optimum, which is validated in our experiments.

Table 2. Parameters of MAD3C used in the experiments.

Parameter	Value
Scheduling actor layer1 size	1024
Scheduling actor layer2 size	300
Scheduling actor learning rate	0.00001
Scheduling critic layer1 size	1024
Scheduling critic layer2 size	300
Scheduling critic learning rate	0.0001
Cooling actor layer1 size	1024
Cooling actor layer2 size	300
Cooling actor learning rate	0.0001
Cooling critic layer1 size	1024
Cooling critic layer2 size	300
Cooling critic learning rate	0.0001
Replay buffer size	1,000,000
Replay start size	10,000
Batch size	64
γ	0.99
τ	0.001
r^{QoS}	0.1
β	0.6
β_1	0.01/300
β_2	1.0
β_3	1.7
β_3	0.3
α^{limit}	5.0

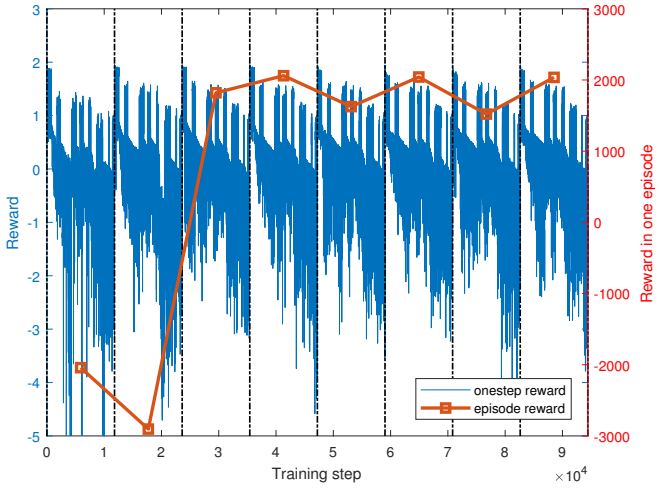
6.3. Results

6.3.1. Convergence

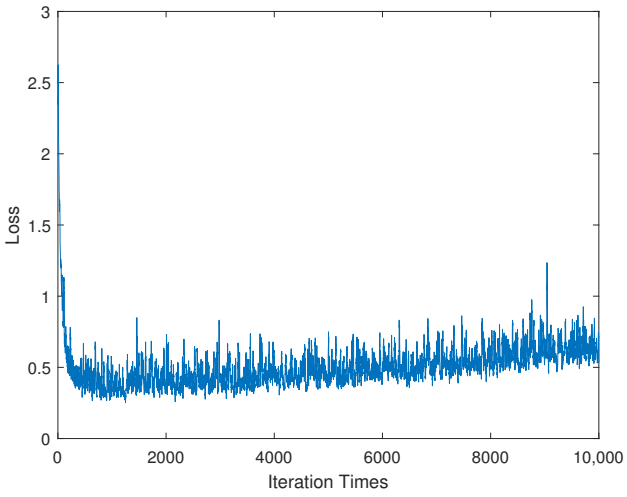
Firstly, the convergence performance of our proposed method MAD3C is validated. Figure 6a shows the reward convergence results over the training steps. The obtained reward for each scheduling and cooling action decision is depicted to show an upward trend in reward values. To show the trend more visually, the sum reward of each episode is also calculated and shown. As we can see in Figure 6a, the agent starts at a low reward decision status first. In the second episode, the reward gets lower because of the random exploration process of the agent. Since the third episode, the algorithm has basically converged, which shows a fast convergence ability. After that, the sum reward keeps stable and relatively high (around 2000), which indicates that MAD3C has converged and stabilized.

In Figure 6b, the loss of the cooling critic in MAD3C is depicted as an example to check the convergence of the algorithm. In the figure, the loss gradually decreases to low values and keeps stable for a long while, which indicates the proposed algorithm MAD3C can achieve a better convergence performance.

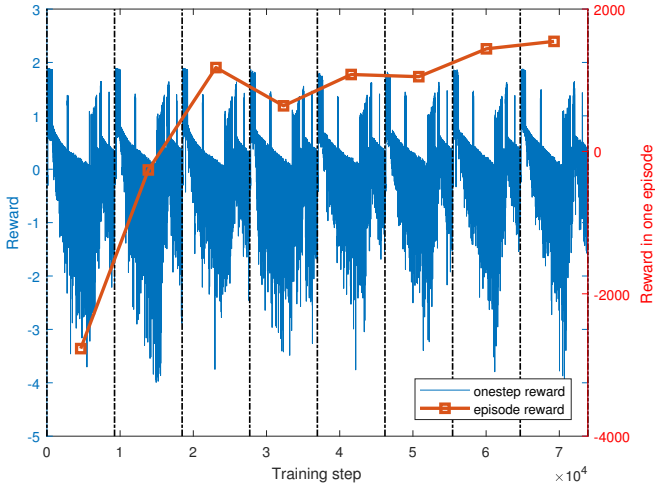
We also validate the algorithm after each training episode. The validation results about reward convergence are presented in Figure 6c. It can be observed that the reward keeps increasing along with the training steps, where the sum reward of each episode is also calculated and revealed. According to the figure, we can verify that the rewards level off and tend to high values in the end, which verifies the convergence and stability of MAD3C.



(**a**) Reward convergence during training.



(**b**) Loss convergence during training.

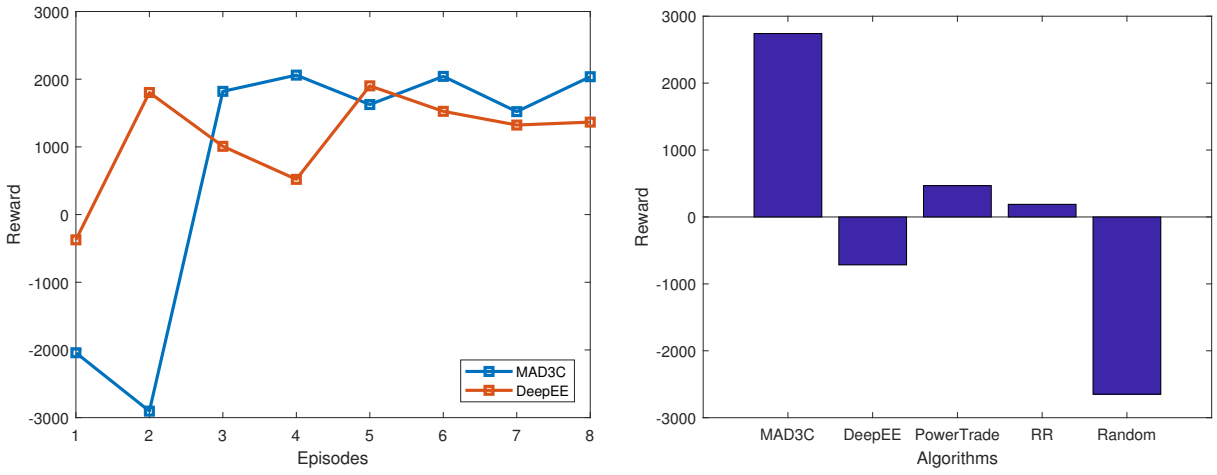


(**c**) Reward convergence during validation.

Figure 6. The training convergence of MAD3C.

6.3.2. Reward

In Figure 7, the reward results during training and test process among different algorithms are presented respectively. Figure 7a shows the reward of MAD3C and DeepEE during training for comparison. Although our method obtains low rewards in the first few training rounds, it converges fast and achieves a high level in the end. In comparison, DeepEE starts at a high sum reward but does not converge in the early stages. At last, DeepEE achieves a relatively lower rewards convergence compared with MAD3C, where the gap between MAD3C and DeepEE is as high as 670.8 (49.1%).



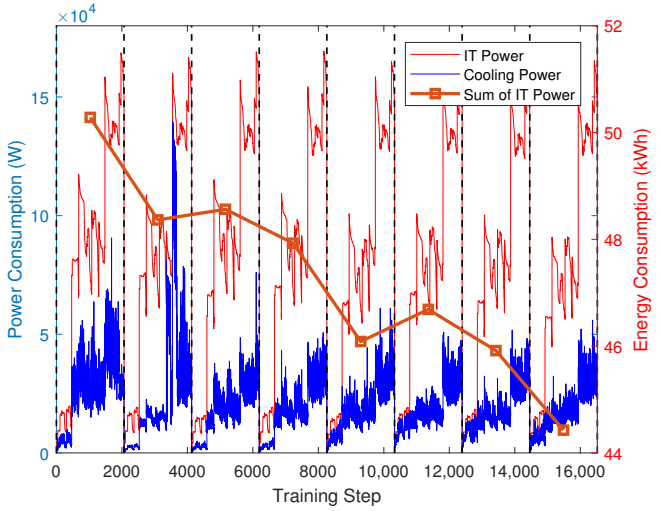
(**a**) Reward comparison during training. (**b**) Reward comparison during test.

Figure 7. The comparison results of reward during training and testing.

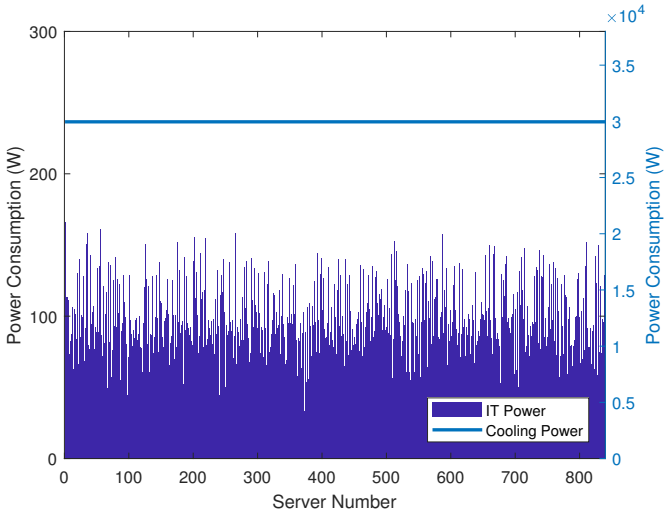
We also compare the episode reward results of MAD3C during the test period with the other four algorithms (DeepEE, PowerTrade, RR and Random), and the results are presented in Figure 7b. According to the figure, we can observe that the reward of a random policy is as low as −2659.2. RR achieves a positive reward, while PowerTrade has a better result compared with RR and Random because of its more advanced joint control strategy. Compared with Random, DeepEE is validated to have learned a policy and achieves a good joint control. However, its reward is still lower than PowerTrade. The main reason is that DeepEE usually causes overloading and thus the reward is deducted. In contrast, it shows that MAD3C outperforms all the algorithms in the test process. It validates that it is effective to cooperatively optimize the energy consumption of IT system and cooling system through MAD3C.

6.3.3. Energy Consumption

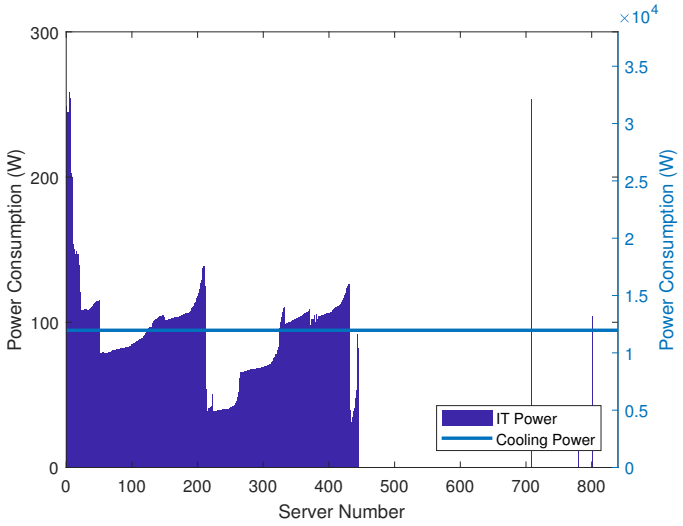
The results of the primary target, energy consumption, are illustrated in Figure 8. Firstly, the power change of the IT system and cooling system at each time slot of each episode of MAD3C is depicted in Figure 8a. It can be seen that the episode cooling power is gradually reduced with the training at a visual rate. Although the IT power consumption varies similarly from episode to episode, by summing the power consumption of one episode (energy consumption), we can find that the IT energy consumption is reduced significantly with training. Specifically, the IT energy consumption during the test is reduced by 11.65%. The cooling energy consumption is reduced by 32.30%. The total energy consumption is reduced by 16.42%. According to the figure, MAD3C shows a stable and reliable energy reduction ability.



(**a**) The power consumption of MAD3C during training.



(**b**) The initial power consumption of the IT system and cooling system.



(**c**) The power consumption of the IT system and cooling system after training.

Figure 8. Energy consumption of MAD3C during training.

To specifically analyze the energy reduction routine of MAD3C, the detailed server power status and cooling status are illustrated in Figure 8. Firstly, Figure 8b presents the initial average power consumption of each server as well as the cooling system. It can be seen that before the training, the agent dispatches the workloads in a random way so that every server needs to be turned on, which would cause more server idle power. The cooling power is generally high since the agent lacks the understanding of the cooling constraints so that the supply air temperature is set to an unnecessary low value and the flow rate is set to a high value. Then, the agent is trained after 8 episodes, and Figure 8c indicates the power consumption of the IT system and cooling system after the training. We find that MAD3C achieves a scheduling algorithm similar to PowerTrade, where a small number of servers run at high load, some servers run at middle load and the rest of the servers split the load equally. In addition, nearly 392 servers keep down so that a large amount of idle power is saved by MAD3C. Meanwhile, a more concrete cooling adjustment strategy is learned by the agent, so that the cooling power is significantly reduced compared with the initial status. Nearly 50% of energy consumption is reduced during test after MAD3C is applied. With the results of Figure 8b,c, it verifies the effectiveness of MAD3C in optimizing the energy consumption of IT systems and cooling systems.

Finally, MAD3C is compared with different control algorithms on the total energy consumption during the test as shown in Figure 9. We can find that the performance of RR and Random is nearly the same, where RR causes more IT energy but less cooling power compared with Random. Meanwhile, PowerTrade significantly reduces the total energy consumption. Compared with PowerTrade, the energy consumption of DeepEE is higher. The main reason we analyze is that DeepEE aims to minimize the PUE ratio but not the total energy consumption, so that more IT energy consumption is wasted as indicated in its target. In comparison, MAD3C is designed for reducing the power consumption of the data center and relieving the heavy burden on the power grid and the environment. As shown in the figure, MAD3C is effective in reducing the total energy consumption compared with DeepEE, PowerTrade, RR and Random. Compared with DeepEE and PowerTrade, 42.82% and 18.95% of energy consumption are reduced by MAD3C, respectively. It validates that the reward design of MAD3C is suitable for the energy reduction target and the architecture design is stable for the eventual performance.

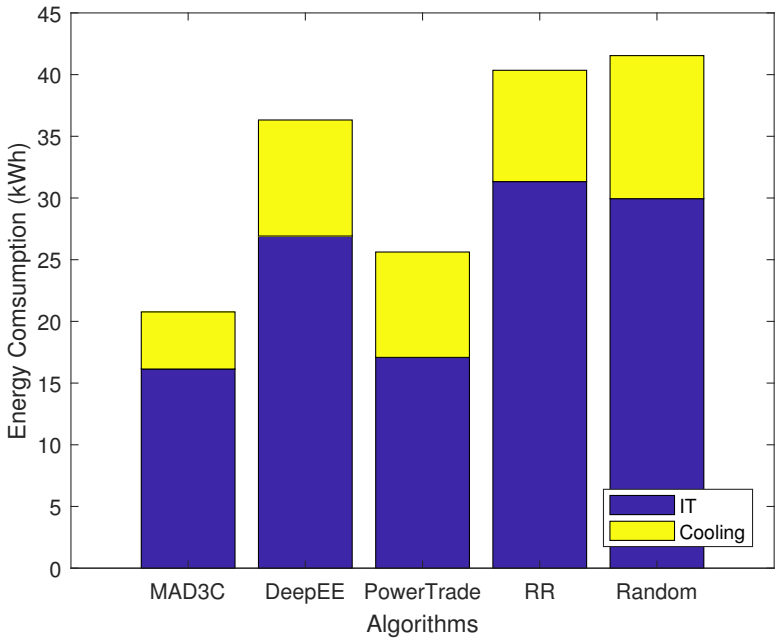


Figure 9. Energy consumption of different algorithms.

6.3.4. Resource Utilization

To validate the effectiveness of AS designed in MAD3C, the average CPU and RAM utilization of active servers is depicted in Figure 10. It can be found that MAD3C can achieve resource utilization even a little higher than the traditional control scheme PowerTrade, which indicates that MAD3C has a well understanding of the resource match between servers and tasks. The eventual resource utilization in DeepEE is slightly higher than Random and RR. Two main reasons cause this situation. Firstly, DeepEE aims at reducing the PUE index, resulting in higher IT power consumption because IT power is used as the denominator in PUE. Secondly, DeepEE has obstacles to extract the resource relationship between servers and tasks. For the second reason, more experiments are conducted in Section 6.3.9 for validations. In summary, MAD3C has an acceptable understanding ability for resource allocation and AS provides a valuable indication for the DRL-based scheduling.

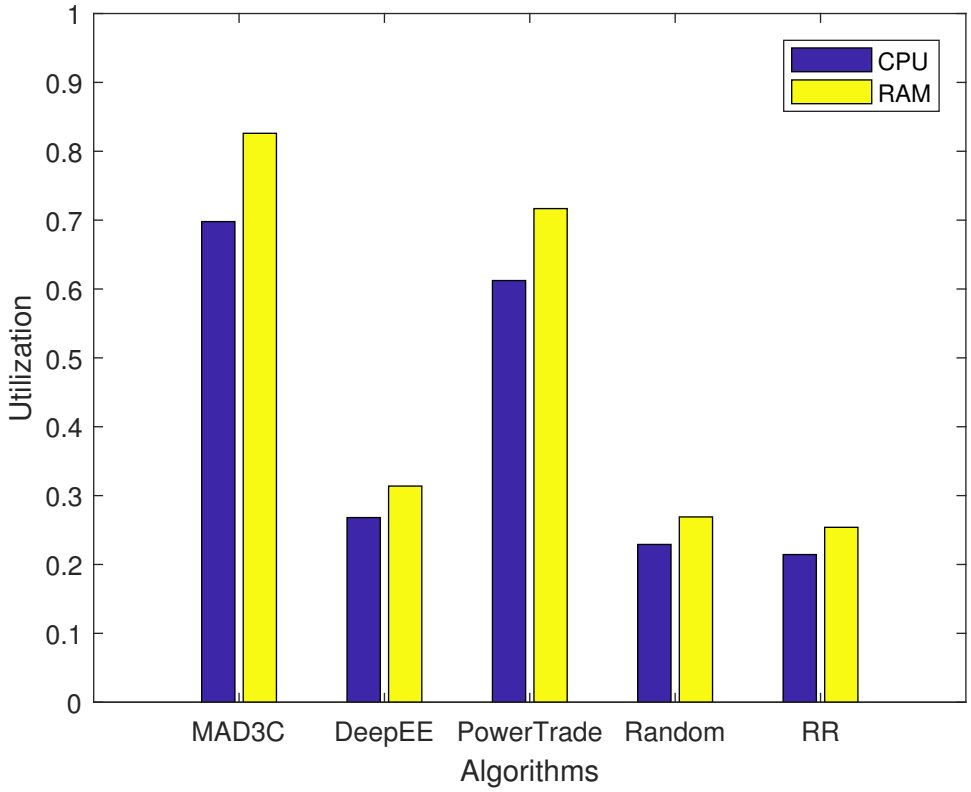


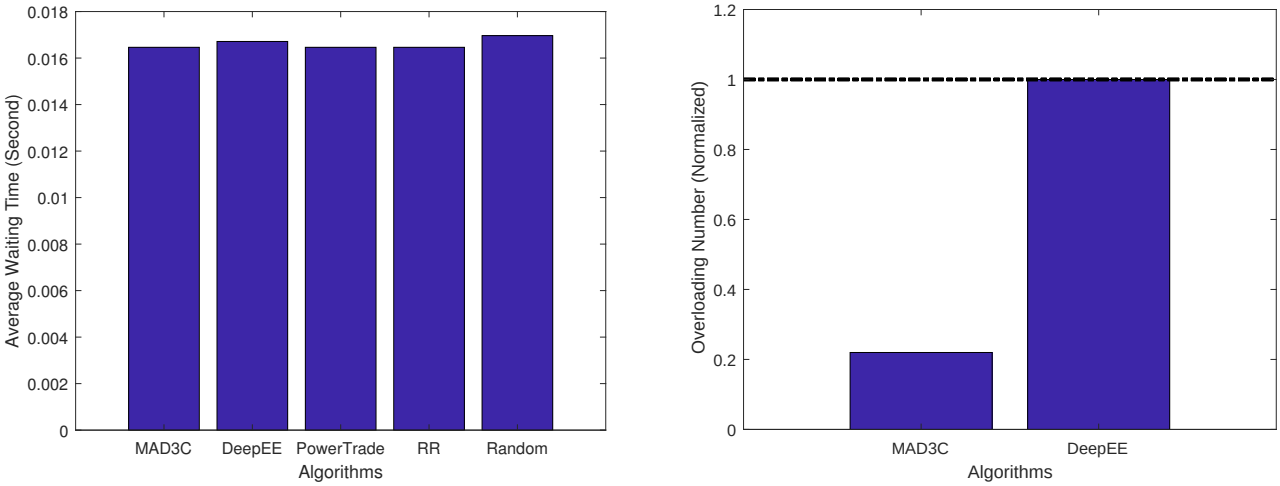
Figure 10. Resource utilization of different algorithms.

6.3.5. QoS

To verify the QoS of the formed scheduling algorithm, the results of task waiting time and the number of overloading servers for each algorithm are collected and depicted in Figure 11a,b, respectively. Firstly, in Figure 11a, the task waiting time of the MAD3C is compared with other baseline algorithms (DeepEE, PowerTrade, RR and Random). As we can see, the task waiting time results of all the algorithms in our experiments are very similar. In combination with Figure 9, we can conclude that MAD3C achieves energy reduction by way of better cooperation of the IT and cooling control rather than at the expense of longer task waiting time. Meanwhile, it also validates the effectiveness of the reward design in MAD3C, where w_k is added to guarantee that the agent will schedule the tasks as soon as possible while taking into account the energy savings.

Secondly, the overloading number result is illustrated in Figure 11b. Since traditional algorithms (PowerTrade, RR) schedule tasks directly to enough-resourced servers, overloading rarely occurs in these algorithms. For MAD3C and DeepEE, the agents need to gradually learn the inter-relationship between the task requested resources and server available resources for energy-efficient scheduling. As soft constraints are applied, over-

loading may occur in these DRL-based scheduling algorithms. However, as illustrated in Figure 11b, MAD3C generates around 21.98% as much overloading servers as DeepEE and keeps a relatively stable state compared with DeepEE. The performance of DeepEE decreases because the multi-dimensional resources are taken into account in our experiments while the NN of DeepEE has difficulty extracting the overloading conditions. In contrast, the relationship between the server resources and the task requested resources is coded directly into the state for the MAD3C agent. Therefore, MAD3C has a better understanding of the resource usage of the servers. As a result, MAD3C outperforms DeepEE at reducing the overloading events as well as reducing the total energy consumption. More importantly, the overloading in MAD3C will be completely eliminated in the end since MAD3C utilizes an overloading protector module in practice, which further guarantees the deployability of MAD3C.

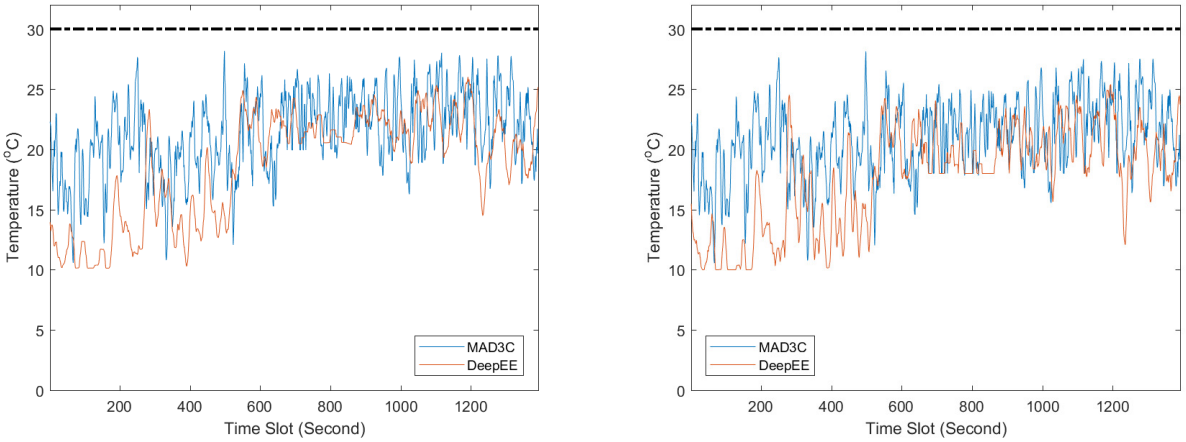


(**a**) The average task waiting time comparison among different algorithms. (**b**) The overloading number of MAD3C and DeepEE.

Figure 11. The comparison results of quality of service (QoS).

6.3.6. Temperature

The next critical evaluation indicator is the temperature in the data center environment. Since PowerTrade and RR utilize model-based cooling control methods, their performance is usually stable. In experiments of temperature, we mainly compare MAD3C with DeepEE to validate the effectiveness of MAD3C. The average outlet air temperature of servers for both approaches under different time slots is illustrated in Figure 12a. According to the figure, it can be observed that the server outlet temperature of both approaches dynamically changes with the time slots. However, MAD3C shows more stability and is closer to the temperature threshold (30 °C), which indicates that MAD3C can save more energy from the cooling system than DeepEE. More specifically, the settings of the supply air temperature of both algorithms are collected and depicted in Figure 12b. It can be seen that the settings of supply air temperature of MAD3C are more stable over time. Furthermore, MAD3C sets a much higher supply air temperature compared with DeepEE but causes no hot spot. It is not because the policy of MAD3C is more aggressive after training, but better cooperation between the IT system and cooling system is achieved in MAD3C so that better workload distribution is realized and thus there is an opportunity to set higher air supply temperatures. Eventually, MAD3C saves more cooling energy consumption than DeepEE. Note that the figure only illustrates the decisions of the agent. When the overheating protector module senses the overheating trends, it will set the supply air temperature to a relatively low value (18 °C in our experiments) so that the servers can always run in a safe mode under the control of MAD3C.



(**a**) The average outlet air temperature.

(**b**) The average supply air temperature.

Figure 12. Outlet and supply air temperature of MAD3C and DeepEE.

6.3.7. Effect of Cooperation

In addition, the influence of the cooperation on the DRL-based control is also evaluated. Three DRL-based control policies are designed for comparison. (1) DRL-Fix utilizes a DRL-based scheduling scheme while the cooling control is fixed. (2) RR-DRL uses RR routine as the scheduling policy and adjusts cooling based DRL algorithm. (3) DRL-DRL adopts DRL for both scheduling and cooling control but has no cooperation between the agents.

Figure 13 depicts the reward changes of all the algorithms during training. It is worth noting that DRL-DRL cannot converge in the end. The main reason is that the agents change their strategies constantly while each agent is unable to capture the patterns of the other agent and its impact on itself. Therefore, it again validates the importance of cooperation for data center energy joint optimization. From the performance of DRL-Fix, it can be concluded that without the coordination of the cooling system, the performance improvement of the DRL-based scheduler is very limited. RR-DRL achieves a relatively high reward in our experiments. The main reason behind it as we analyze is that the RR policy rarely causes overloading penalty while the workload is evenly distributed so that the cooling adjustment is benefited to achieve low energy consumption. Even so, the reward gap between MAD3C and RR-DRL is as high as 1130.3 (123.68%). MAD3C achieves the highest sum reward and keeps stable in the end. Therefore, the cooperation between the IT system and cooling system is crucial for energy reduction for data centers.

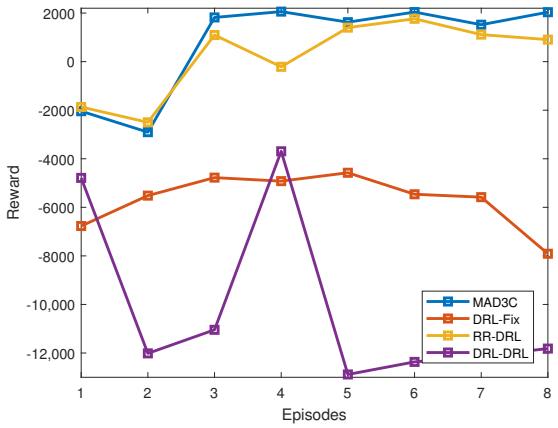


Figure 13. Reward of MAD3C, DRL-Fix, Round Robin (RR)-DRL, DRL-DRL during training.

6.3.8. Effect of Scheduling Baseline Comparison Method

Another critical feature of MAD3C is the scheduling baseline comparison, which is used to stabilize the DRL architecture. We try to use different baseline algorithms for our DRL model. (1) MAD3C-CSF is the leading implementation of MAD3C where the Coolest Server First scheme is utilized as the baseline algorithm. (2) MAD3C-RR adopts the RR scheme to build the DRL model. (3) MAD3C-none utilizes no baseline algorithm, which is trained as a comparison.

We compare the reward training results along with the training steps for algorithm MAD3C-CSF, MAD3C-RR and MAD3C-none. As shown in Figure 14, it can be found that both MAD3C-CSF and MAD3C-RR obtains high reward in the end. MAD3C-CSF has a faster convergence ability and achieves a little higher reward than MAD3C-RR. It is reasonable since CSF contains more information compared with the RR scheme for the DRL, which is conducive for the training of agents. On the other hand, MAD3C-none fails to converge eventually. It validates the effectiveness and significance of the existence of scheduling baseline comparison method in the MAD3C architecture as our analysis.

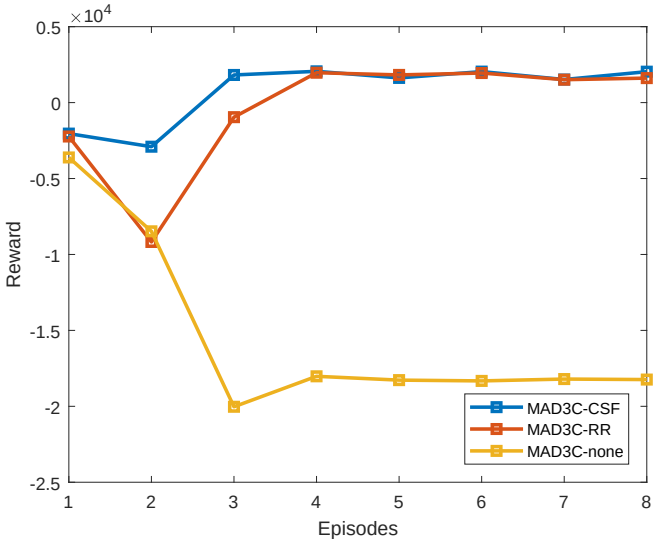
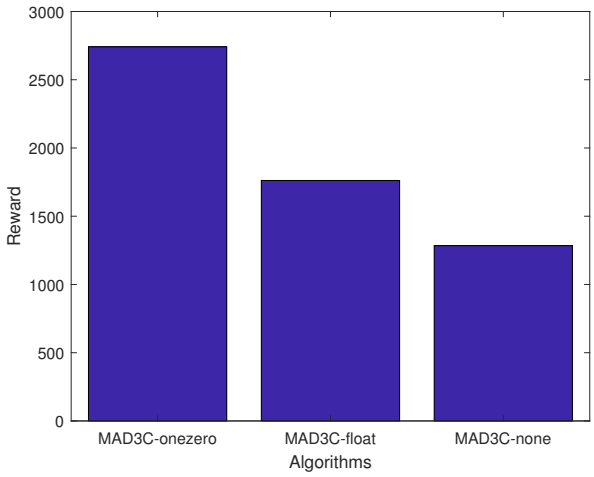


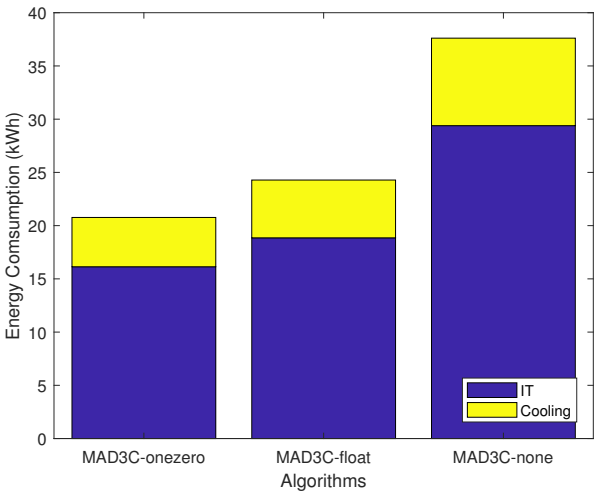
Figure 14. Reward comparison among MAD3C-CSF, MAD3C-RR and MAD3C-none during training.

6.3.9. Effect of Adaptive Score

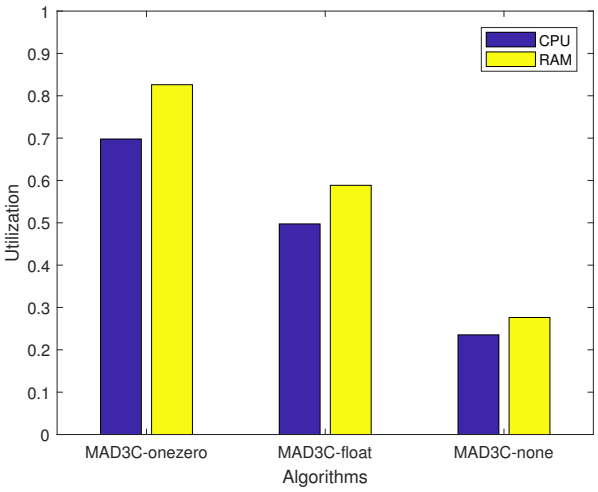
The third critical factor which may affect the DRL performance is the adaptive score designed in MAD3C. Therefore, different adaptive score functions are evaluated to verify the influence of AS. As introduced in Section 4.2.1, an integer model and a float model of AS are specified, based on which the algorithms are named as MAD3C-onezero and MAD3C-float in the experiments. Moreover, MAD3C without an AS is also evaluated, named as MAD3C-none. The results of test reward, energy consumption are shown in Figure 15a,b, respectively. Firstly, MAD3C-onezero outperforms the other two mechanisms and achieves the highest reward. Therefore, the AS scheme is valuable for the performance improvement of MAD3C. It is unfortunate to see that MAD3C-float performs a little worse than MAD3C-onezero in reward and energy consumption. The main reason according to our analysis is that the soft representation of AS may still sometimes mislead the agent when trading off the multiple goals and intensifies the difficulty for agents to understand the environment. Even so, MAD3C-float obtains higher rewards and lower energy consumption compared with MAD3C-none. In addition, as shown in Figure 15c, MAD3C-float achieves higher resource utilization than MAD3C-none, while MAD3C-onezero has the best performance of improving the resource utilization. It validates that the design of AS is beneficial to the resource utilization improvement of MAD3C and the direct integer AS representation (Equation (8)) has a better effect as shown in our experiments.



(**a**) The reward of MAD3C-onezero, MAD3C-float and MAD3C-none during test.



(**b**) The energy consumption of MAD3C-onezero, MAD3C-float and MAD3C-none during test.



(**c**) The resource utilization results of MAD3C-onezero, MAD3C-float and MAD3C-none during test.

Figure 15. The comparison results of MAD3C-onezero, MAD3C-float and MAD3C-none.

6.4. Discussion

According to the experiment evaluation, MAD3C shows a considerable performance in data center energy savings. The task migration can also be integrated into our method design. For instance, a VM migration strategy is implemented in the well-known OpenStack platform, which migrates VMs for aggregation when resource utilization is low, thus generating more idle servers and turning them off. Our method mainly focuses on task placement when tasks first arrive. After the first allocation, if there exist servers with low utilization for a long period, it indicates that the DRL agents may miscalculate the future outcomes (Q-value). In this case, we can still interfere with the DRL's decisions and utilize VM migration strategies like OpenStack for workload consolidation to save more energy, which is beneficial for the actual deployment of DRL-based control approaches. However, it should be noted that the used workload consolidation strategy should still meet the joint energy consumption optimization goal; that is, the cooling power consumption should not increase dramatically after the workload consolidation.

Although MAD3C has shown an effective performance on the joint energy optimization in data centers, there are still some limitations that need to be solved in future work. First, system security becomes critical for data centers if the cooling control authority is transferred to software. For instance, if a hacker gains access to the cooling control, the supply air temperature and flow rate may be adjusted to infeasible values, leading to extensive server overheating and damage. Therefore, in order to achieve joint optimization control in real data centers, security needs to be guaranteed first. Second, the frequency of information collection in data centers such as power and temperature is usually at a minute level. It is unable to match the task arrival frequency, which may be on the order of seconds or even microseconds. Training the DRL agents to adapt to the environment with delayed information will be difficult and valuable for DRL-based energy reduction approaches in data centers. Third, the poor interpretability of DRL NNs also limits the deployment of DRL-based scheduling and cooling control methods in data centers. Moreover, the DRL's consideration of future long-term returns further worsens the interpretability of its decisions. As a result, it is difficult to evaluate the rationality of DRL decisions in real time and to use them directly. Improving the interpretability of DRL-based scheduling and cooling control approaches is meaningful for their application in practice. In future work, we intend to develop an interpretable DRL-based joint energy efficiency optimization approach that can handle the information delay issue, and we will evaluate the performance in a real testbed.

7. Conclusions

To efficiently improve the energy efficiency of data centers, joint optimization of the IT system and cooling system has become an effective and even indispensable way. However, due to the inadequate models and the high-dimensional state-action spaces, the existing methods are difficult to achieve an expected performance as theoretical analysis. Although some DRL-based optimization approaches solve the problems, they still cannot sufficiently handle the joint control of IT and cooling system, and thus better cooperation is demanded. Meanwhile, the multi-dimensional resources should be taken into account to further optimize the server resources and improve data center energy efficiency. In this paper, in order to solve the challenges, MAD3C is presented based on model-free DRL algorithms. To further improve the energy efficiency of data centers, we focus on improving the cooperation between the IT system and cooling system, so a hybrid AC-DDPG cooperative framework is designed in this paper based on multi-agent DRL methods. Through the joint consideration of both scheduling action and cooling action in each critic in the framework, the cooperation of IT and cooling systems is achieved. Meanwhile, two latent issues including the instability and high calculation overhead that exist in the framework are analyzed and solved by our proposed scheduling baseline comparison method. In addition, an adaptive score scheme is designed for the architecture in consideration of multi-dimensional resources and resource utilization improvement.

Finally, experiments are conducted and validate that our approach is effective in reducing more energy for data centers through the cooperation design while the training stability is guaranteed and the resource utilization is verified to be improved.

Author Contributions: Conceptualization, C.C. and Z.L.; methodology, C.C. and F.Z.; software, C.C. and P.S.; validation, K.J. and S.Z.; formal analysis, C.C.; investigation, K.J. and A.M.; resources, K.J., P.S. and A.M., D.Q.; data curation, K.J., P.S. and A.M.; writing—original draft, C.C.; writing—review and editing, F.Z. and Z.L.; visualization, C.C.; supervision, Z.L.; project administration, F.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Key Research and Development Program of China (grant numbers 2017YFB1010001); the National Natural Science Foundation of China (grant numbers 61520106005, 61761136014). The corresponding author is Zhiyong Liu (zyliu@ict.ac.cn).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable. No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shehabi, A.; Smith, S.; Sartor, D.; Brown, R.; Herrlin, M.; Koomey, J.; Masanet, E.; Horner, N.; Azevedo, I.; Lintner, W. *United States Data Center Energy Usage Report*; Available online: https://escholarship.org/content/qt84p772fc/qt84p772fc.pdf (accessed on 1 January 2021).
2. Jones, N. How to stop data centres from gobbling up the world's electricity. *Nature* **2018**, *561*, 163–167. [CrossRef]
3. Zhang, W.; Wen, Y.; Wong, Y.W.; Toh, K.C.; Chen, C.H. Towards joint optimization over ICT and cooling systems in data centre: A survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1596–1616. [CrossRef]
4. Gu, C.; Liu, C.; Zhang, J.; Huang, H.; Jia, X. Green scheduling for cloud data centers using renewable resources. In Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 26 April–1 May 2015; pp. 354–359.
5. Xu, X.; Dou, W.; Zhang, X.; Chen, J. EnReal: An energy-aware resource allocation method for scientific workflow executions in cloud environment. *IEEE Trans. Cloud Comput.* **2015**, *4*, 166–179. [CrossRef]
6. Xu, X.; Zhang, X.; Khan, M.; Dou, W.; Xue, S.; Yu, S. A balanced virtual machine scheduling method for energy-performance trade-offs in cyber-physical cloud systems. *Future Gener. Comput. Syst.* **2020**, *105*, 789–799. [CrossRef]
7. Babu, G.P.; Tiwari, A. Energy Efficient Scheduling Algorithm for Cloud Computing Systems Based on Prediction Model. *Int. J. Adv. Netw. Appl.* **2019**, *10*, 4013–4018. [CrossRef]
8. Zhou, R.; Wang, Z.; Bash, C.E.; McReynolds, A.; Hoover, C.; Shih, R.; Kumari, N.; Sharma, R.K. A holistic and optimal approach for data center cooling management. In Proceedings of the 2011 American Control Conference, San Francisco, CA, USA, 29 June–1 July 2011; pp. 1346–1351.
9. Beghi, A.; Cecchinato, L.; Dalla Mana, G.; Lionello, M.; Rampazzo, M.; Sisti, E. Modelling and control of a free cooling system for data centers. *Energy Procedia* **2017**, *140*, 447–457. [CrossRef]
10. Wang, Q.; Yu, Y.; Li, B.; Zhu, Y. Tensor-Based Optimal Temperature Control of CRACs in Multi-Datacenters. *IEEE Access* **2019**, *7*, 41445–41453. [CrossRef]
11. Pakbaznia, E.; Pedram, M. Minimizing data center cooling and server power costs. In Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design, San Fancisco, CA, USA, 19–21 August 2009; pp. 145–150.
12. Ahmad, F.; Vijaykumar, T. Joint optimization of idle and cooling power in data centers while maintaining response time. *ACM Sigplan Not.* **2010**, *45*, 243–256. [CrossRef]
13. Parolini, L.; Sinopoli, B.; Krogh, B.H.; Wang, Z. A cyber–physical systems approach to data center modeling and control for energy efficiency. *Proc. IEEE* **2011**, *100*, 254–268. [CrossRef]
14. Li, S.; Le, H.; Pham, N.; Heo, J.; Abdelzaher, T. Joint optimization of computing and cooling energy: Analytic model and a machine room case study. In Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems, Macau, China, 18–21 June 2012; pp. 396–405.
15. Al-Qawasmeh, A.M.; Pasricha, S.; Maciejewski, A.A.; Siegel, H.J. Power and thermal-aware workload allocation in heterogeneous data centers. *IEEE Trans. Comput.* **2013**, *64*, 477–491. [CrossRef]
16. Chen, T.; Wang, X.; Giannakis, G.B. Cooling-aware energy and workload management in data centers via stochastic optimization. *IEEE J. Sel. Top. Signal Process.* **2015**, *10*, 402–415. [CrossRef]

17. Wang, Y.; Zhang, F.; Wang, R.; Shi, Y.; Guo, H.; Liu, Z. Real-time Task Scheduling for joint energy efficiency optimization in data centers. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017; pp. 838–843.
18. Wan, J.; Gui, X.; Zhang, R.; Fu, L. Joint cooling and server control in data centers: A cross-layer framework for holistic energy minimization. *IEEE Syst. J.* **2017**, *12*, 2461–2472. [CrossRef]
19. Ran, Y.; Hu, H.; Zhou, X.; Wen, Y. DeepEE: Joint Optimization of Job Scheduling and Cooling Control for Data Center Energy Efficiency Using Deep Reinforcement Learning. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 645–655.
20. Mao, H.; Alizadeh, M.; Menache, I.; Kandula, S. Resource management with deep reinforcement learning. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks, Atlanta, GA, USA, 14–15 November 2016; pp. 50–56.
21. Yi, D.; Zhou, X.; Wen, Y.; Tan, R. Toward efficient compute-intensive job allocation for green data centers: A deep reinforcement learning approach. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–9 July 2019; pp. 634–644.
22. Hu, Z.; Tu, J.; Li, B. Spear: Optimized Dependency-Aware Task Scheduling with Deep Reinforcement Learning. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 2037–2046.
23. Moriyama, T.; De Magistris, G.; Tatsubori, M.; Pham, T.H.; Munawar, A.; Tachibana, R. Reinforcement Learning Testbed for Power-Consumption Optimization. In Proceedings of the Asian Simulation Conference, Hong Kong, China, 3–5 December 2018; pp. 45–59.
24. Li, Y.; Wen, Y.; Tao, D.; Guan, K. Transforming cooling optimization for green data center via deep reinforcement learning. *IEEE Trans. Cybern.* **2019**, *50*, 2002–2013. [CrossRef] [PubMed]
25. Rjoub, G.; Bentahar, J.; Abdel Wahab, O.; Saleh Bataineh, A. Deep and Reinforcement Learning for Automated Task Scheduling in Large-Scale Cloud Computing Systems. 2020; p. e5919. Available online: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5919 (accessed on 1 January 2021)
26. Chi, C.; Ji, K.; Marahatta, A.; Song, P.; Zhang, F.; Liu, Z. Jointly optimizing the IT and cooling systems for data center energy efficiency based on multi-agent deep reinforcement learning. In Proceedings of the Eleventh ACM International Conference on Future Energy Systems, Melbourne, Australia, 22–26 June 2020; pp. 489–495.
27. Fan, X.; Weber, W.D.; Barroso, L.A. Power provisioning for a warehouse-sized computer. *ACM Sigarch Comput. Archit. News* **2007**, *35*, 13–23. [CrossRef]
28. Kong, F.; Liu, X. A survey on green-energy-aware power management for datacenters. *ACM Comput. Surv. (CSUR)* **2014**, *47*, 1–38. [CrossRef]
29. Liu, Z.; Chen, Y.; Bash, C.; Wierman, A.; Gmach, D.; Wang, Z.; Marwah, M.; Hyser, C. Renewable and cooling aware workload management for sustainable data centers. In Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, London, UK, 11–15 June 2012; pp. 175–186.
30. Zhao, Z.; Wu, F.; Ren, S.; Gao, X.; Chen, G.; Cui, Y. Tech: A thermal-aware and cost efficient mechanism for colocation demand response. In Proceedings of the 2016 45th International Conference on Parallel Processing (ICPP), Philadelphia, PA, USA, 16–19 August 2016; pp. 464–473.
31. Marahatta, A.; Pirbhulal, S.; Zhang, F.; Parizi, R.M.; Choo, K.K.R.; Liu, Z. Classification-based and energy-efficient dynamic task scheduling scheme for virtualized cloud data center. *IEEE Trans. Cloud Comput.* **2019**. [CrossRef]
32. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
33. Konda, V.R.; Tsitsiklis, J.N. Actor-critic algorithms. In Proceedings of the 12th International Conference on Neural Information Processing Systems, Denver, CO, USA, 29 November–4 December 1999.
34. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
35. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, O.P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Long Beach, CA, USA, 2017; pp. 6379–6390.
36. Foerster, J.N.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
37. Tang, Q.; Mukherjee, T.; Gupta, S.K.; Cayton, P. Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters. In Proceedings of the 2006 Fourth International Conference on Intelligent Sensing and Information Processing, Bangalore, India, 15 October–18 December 2006; pp. 203–208.
38. Reiss, C.; Wilkes, J.; Hellerstein, J.L. *Google Cluster-Usage Traces: Format+ Schema*; White Paper; Google Inc.: Mountain View, CA, USA, 2011; pp. 1–14.