Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Reinforcement learning based task scheduling for environmentally sustainable federated cloud computing

Zhibao Wang[1,2], Shuaijun Chen[1], Lu Bai[3*], Juntao Gao[1], Jinhua Tao[4], Raymond R. Bond[5] and Maurice D. Mulvenna[5]

Abstract

The significant energy consumption within data centers is an essential contributor to global energy consumption and carbon emissions. Therefore, reducing energy consumption and carbon emissions in data centers plays a crucial role in sustainable development. Traditional cloud computing has reached a bottleneck, primarily due to high energy consumption. The emerging federated cloud approach can reduce the energy consumption and carbon emissions of cloud data centers by leveraging the geographical differences of multiple cloud data centers in a federated cloud. In this paper, we propose Eco-friendly Reinforcement Learning in Federated Cloud (ERLFC), a framework that uses reinforcement learning for task scheduling in a federated cloud environment. ERLFC aims to intelligently consider the state of each data center and effectively harness the variations in energy and carbon emission ratios across geographically distributed cloud data centers in the federated cloud. We build ERLFC using Actor-Critic algorithm, which select the appropriate data center to assign a task based on various factors such as energy consumption, cooling method, waiting time of the task, energy type, emission ratio, and total energy consumption of the current cloud data center and the details of the next task. To demonstrate the effectiveness of ERLFC, we conducted simulations based on real-world task execution data, and the results show that ERLFC can effectively reduce energy consumption and emissions during task execution. In comparison to Round Robin, Random, SO, and GJO algorithms, ERLFC achieves respective reductions of 1.09, 1.08, 1.21, and 1.26 times in terms of energy saving and emission reduction.

Keywords Cloud computing, Federated cloud, Reinforcement learning, Energy efficiency, Carbon emissions, Task scheduling

Introduction

Cloud computing [1] has become the most popular computing paradigm today for distributed computing and parallel processing [2] because of its elasticity and scalability and pay-per-use model [3]. Despite its popularity, traditional cloud computing paradigm has reached a plateau, exposing many limitations such as resource contention, service interruption, lack of interoperability in data representation, degradation of Quality of Service (QoS), and especially high energy consumption and carbon emissions [4–8]. The enormous pressure on energy consumption and carbon emission [9] severely hampers the continued development of cloud data centers. Due to

*Correspondence:
Lu Bai
l.bai@qub.ac.uk
[1] School of Computer and Information Technology, Northeast Petroleum University, Daqing 163318, China
[2] Bohai-Rim Energy Research Institute, Northeast Petroleum University, Qinhuangdao 066004, China
[3] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT9 6SB, UK
[4] State Key Laboratory of Remote Sensing Science, Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing Normal University, Beijing 100101, China
[5] School of Computing, Ulster University, Belfast BT15 1ED, UK

the serious challenge of global warming, green computing [10, 11] is gaining more popularity. In China, energy consumption and CO2 emissions increased by 70.97% and 47.15%, respectively, from 82.88 exajoules and 6.673 billion tons in 2006 to 14,170 exajoules and 982.58 billion tons in 2019 [12]. In Saudi Arabia, the total emissions of greenhouse gas from electricity consumption increased from 58.2 million tons in 1971 to 58.88 million tons in 2020, accounting for 1.45% of global carbon emissions and the total emissions from electricity in Saudi Arabia are increasing to an average annual rate of more than 5% [13]. According to the US Environment Protection Agency (EPA), there were nearly 44 million servers worldwide in 2007, which consumed approximately 0.5% of the world's electricity, and U.S. data centers consume as much electricity as five power plants in a year [14]. Their data centers also produced 80 metric tons of carbon emissions per year, equivalent to the carbon footprint of the Netherlands and Argentina combined [15]. Therefore, the information and communication technology sector is identified as a significant contributor to global energy consumption and greenhouse gas emissions, and data centers are projected to account for 13% of global energy consumption by 2030 due to the increased energy demand from data centers [16].

Many solutions have been proposed by both industry and academia to address the energy consumption dilemma faced by cloud computing. Among these solutions, federated clouds [17, 18] perform the best and are most likely to be the key solution to the problems associated with cloud computing. A federated cloud is a collaborative network of resources from multiple cloud service providers (CSPs). Each CSP in the federated cloud can forward the request to the cloud data center of the CSP that is best suited for computing based on the characteristics of the user request. This approach optimizes energy consumption and carbon emissions by taking advantage of the differences in energy sources between the geographic locations of different data centers. Additionally, as the federated cloud aggregates the resources of multiple CSPs, it reduces the need for hardware resources from individual CSPs, thereby reducing their operational costs. The federated cloud architecture presents opportunities for small and medium sized CSPs, enhancing their competitiveness with larger CSPs [19].

One of the most complex challenges in a federated cloud environment is to optimize the diverse resources offered by various CSPs. This involves making informed decisions about task allocations, specifically determining the most suitable data center and server room for executing tasks. The aim is to minimize energy consumption and carbon emissions while satisfying the user's QoS. The geographical location of different cloud data centers results in different energy types and varying carbon emissions even for the same amount of energy consumption. Additionally, the current workload of each server in the federated cloud can vary, which adds to the complexity of this issue. Cloud data centers are composed of IT equipment such as servers, network equipment, storage equipment and infrastructure equipment such as cooling and power equipment. Cooling equipment and servers are considered to be the main contributors to the energy consumption of cloud data centers [20–22], accounting for 80% of the total energy consumption of data centers [23]. Server energy consumption and cooling energy consumption are interdependent, and when server energy consumption rises, the cooling energy consumption also rises in tandem. Energy consumption of servers can be divided into two types; one is static energy consumption, i.e., the standby energy consumption of servers. The other is dynamic energy consumption, i.e., the energy consumption of the server when computing tasks. Data center cooling methods include air cooling and liquid cooling [24], with each method generating different energy consumption. To reduce the energy consumption of cloud data centers, it is necessary to consider these two energy consumptions. The scheduling algorithm should consider multiple metrics to make an intelligent decision on how to reduce the energy consumption of both the cooling method and the energy consumption of the servers requires.

A task is presented as a directed acyclic graph (DAG) where each vertex of the DAG represents a subtask in the task, and the edges between different vertices represent the dependencies between different subtasks. This dependency relationship makes it easy to describe the parallelism of the task. Usually, data transfer between different subtasks results in a large amount of energy consumption [25]. To reduce the energy consumption of data transfer between different tasks, we propose to allocate all subtasks in the same DAG to the same cloud data center and machine room for computation. To achieve this, we use reinforcement learning to determine the optimal assignment of tasks to specific server rooms within cloud data centers. Then we assign the task to the corresponding server for computation using Heterogeneous Earliest Finish Time (HEFT) algorithm [26], which helps reduce the energy consumption and emissions of the data center. To make the decision of the scheduling algorithm more detailed, we consider the load situation, cooling method, total energy consumption, total emissions, emission ratio, and the next task details for each server room in each data center. This helps the scheduling algorithm choose the most suitable server for executing the task, and further reducing energy consumption and emissions. Furthermore, this paper introduces the

watermark to simulate the randomness of task arrival time to create a more relevant task scheduling environment. To better cope with the real-time scheduling, we aim to minimize the complexity of the neural network architecture, enabling the scheduling algorithm to make quicker decisions upon task arrival.

The main contribution of this paper is summarized as follows:

- A suitable reinforcement learning framework (ERLFC) is developed for fast task allocation decisions in a federated cloud environment, leading to reduced energy consumption and carbon emissions.
- The ERLFC is compared with classical task scheduling algorithms and metaheuristics algorithms, and our experimental results show that the ERLFC still has a strong competitive edge even in scenarios with large task volumes.
- We take full advantage of the federated cloud and the differences of geographical locations of CSPs by introducing six different energy types and diverse carbon emission ratios to provide novel insights for energy consumption and carbon emissions reduction.
- Diverse cooling methods are introduced to further reduce energy consumption and carbon emissions by optimizing the control of cooling methods.

The remaining of this paper is organized as follows. Section 2 provides a review of the relevant works. The proposed system model is presented in Sect. 3. Experimental details, the results and discussions are presented in Sect. 4. Section 5 concludes the paper.

Related work

In recent years, the energy consumption and emission of cloud data centers have become the main limitation for the development of cloud computing. Scholars have conducted extensive research and exploration in energy saving and emission reduction techniques. These research efforts have been divided into four main categories, which will be discussed in detail in the following subsections.

Algorithms based on heuristics

Yuan et al. [7] proposed an adaptive simulated-annealing-based biobjective differential evolution (ASBD) algorithm that takes full advantage of the variation in power prices brought about by the diversity of different cloud data geographic locations to reduce the average cost of energy. This approach minimizes the energy cost as well as the average task loss of the cloud data center by constructing a Pareto-optimal set using the minimum Manhattan distance approach from the number of data center energized servers and the portal task allocation data to decide on an optimal solution. Another approach, proposed by Hogade et al. [27] is to use three workload management mechanisms that take into account factors such as data center cooling power, co-location interference, time-of-use tariff, renewable energy, net metering, and peak demand pricing to reduce data center energy consumption and cost based on geographic load. In [28], Ben Alla et al. proposed an efficient deadline and energy-aware task scheduling algorithm to improve efficiency and reduce the consumption of cloud resources under the deadline constraints. The algorithm optimizes the scheduling service of any tasks in terms of makespan, energy consumption, resource utilization, and load balancing. However, many of the parameters in this algorithm are pre-set, such as the number of instructions for the task, which can be challenging to estimate in real applications. In [29], the assignment of tasks is abstracted into the deployment of virtual machines, and by estimating the future workload and the current resource distribution. The resources are allocated using a simulated annealing algorithm to balance the load among servers to reduce the power consumption in cloud computing systems. While heuristic algorithms, such as this one, have lower computational cost and require only information about the current task, they may not be well-suited for real-time environments because of their high response delay in time-sensitive scenarios.

Algorithms based on dynamic voltage and frequency scaling

Xie et al. [30] proposed an energy efficient way to recover idle time by introducing the concept of latest completion time through dynamic voltage and frequency energy efficient design optimization techniques. In the case of meeting the task deadline constraint, the dynamic energy consumption is reduced by moving the task to the idle area that can generate the least dynamic energy processor, thus reducing the overall energy consumption. In [31], Safari and Khorsand introduced a new energy-efficient scheduling method for time-limited workflow tasks using dynamic voltage and frequency scaling techniques. This method reduces energy consumption by adjusting the voltage according to the machine's frequency, operating and adjusting the frequency according to the current host usage. Tang et al. [32] proposed a workflow task scheduling algorithm based on dynamic voltage and frequency scaling techniques. The algorithm primarily focuses on allocating tasks to computing resources that are relatively available, considering their deadlines. Additionally, it intelligently shuts down the current computing resources to enhance resource utilization,

thereby achieving energy-saving benefits. Baskiyar and Abdel-Kader [33] proposed a scheduling algorithm on a heterogeneous processor running at discrete operating voltages, which exploits the dependencies between tasks to achieve energy savings by dynamically scaling the voltage of the processor. The algorithm focuses on converting the DAG into a single-entry and single-exit DAG, and dynamically scales the CPU voltage by calculating the difference between the end time of tasks on the non-critical path and tasks on the critical path. A joint multi-objective optimization of task scheduling and energy consumption based on dynamic voltage and frequency scaling (DVFS) technique and whale optimization algorithm [34] was proposed in [35] to reduce energy consumption from device overhead by scheduling tasks flexibly in mobile cloud computing. DVFS enables tasks to run at reduced voltage and clock frequency to fill idle time and reduce energy consumption. However, DVFS often requires a high degree of coupling between tasks and resources [36] and its transitions are too long, leading to reduced response time for tasks and inability to meet the requirements of real-time tasks. Additionally, DVFS is only applicable to CPUs, which consume only a small fraction of the total system power, leaving little room for DVFS in future technologies [37].

Algorithms based on joint optimization and adaptive systems

Wang et al. [38] proposed a real-time task classification and scheduling strategy. This approach involves classifying real-time tasks and assigning tasks with similar execution times and end times to the same server, ultimately enhancing server utilization. This results in optimizing the energy efficiency of the cooling system and the server as a way to reduce energy consumption. A joint energy efficiency optimization scheme based on marginal cost and task grouping is proposed in [39], considering the cooling energy consumption that is generated by different cooling methods and uses these according to the characteristics of the tasks to achieve energy savings. A joint optimization scheme is proposed in [37], which reduces the energy consumption by reducing the sum of idle power and cooling power, decreasing the response time, and reducing the total power and local power hotspots. Jiang et al. [40] proposed an adaptive scheme to schedule a heterogeneous task with random arrivals, reducing the energy consumption of mobile devices. The algorithm is based on a continuous-time Markov decision process that formulates energy-efficient and QoS-guaranteed task scheduling as a constrained stochastic optimization problem. Chase et al. [41] proposed a resource allocation strategy that improves the energy efficiency of server clusters by balancing the cost of resources and the benefits derived from their use. The implementation concentrates the load of the data center on a subset of servers and saves energy by shutting down the remaining servers.

Algorithms based on reinforcement learning and neural networks

Deep reinforcement learning has been widely used in the domain of cloud job scheduling. The recent research by Zhang et al. [42] introduces a novel cost-aware scheduling system for real-time workflows in the cloud by combining the global search capability of genetic algorithm and the environment awareness decision-making capability of deep reinforcement learning techniques. Cheng et al. [43] proposed an approach to enhance scheduling policy training with efficient preemptive mechanisms to minimize job execution costs and meet user response time expectations. Cheng et al. [44] introduced a deep reinforcement learning based method for real-time job scheduling in hybrid clouds, prioritizing cost-effective job execution without compromising quality of service and minimizing response time. Ding et al. [45] proposed a Q-learning based energy-efficient cloud computing task scheduling, which aims to reduce energy consumption by using Q-learning policies to compute the most suitable virtual machines for task execution. This reduces task response time and maximizes CPU utilization per server. Similarly, Siddesha et al. [46] proposed a Q-learning based task scheduling approach that considers the load balancing capability of VMs and introduces a novel strategy to enhance the reward of task scheduling for better resource sharing and reduced duration of computational policies. However, Q-learning for task scheduling decisions can be challenging because the state of cloud data centers is constantly changing making it difficult to maintain an infinite state. Yan et al. [47] proposed a strategy to model data center temperature and cooling energy consumption using neural networks to compute the optimal placement of servers by genetic algorithms. The main objective is to minimize the maximum temperature of the data center server room and minimize the minimum power of new servers to achieve energy saving. This method mainly considers the energy consumption in the data center server room and does not address large scale energy savings. Kang et al. [48] proposed an adaptive DRL-based constraint framework for energy-aware task scheduling with changing workloads and unpredictable nature. The framework consists of three parts, the first part detects and predicts future workload changes, the second part adjusts the discount factor to determine agent changes, and the third part uses the deep Q-network algorithm to achieve task scheduling. Although the above task scheduling algorithms consider energy consumption, they do not address the issue of emissions.

Having reviewed the related works, we found that previous efforts to achieve energy efficiency in cloud environments focused on single clouds and hybrid clouds. However, due to limitations in geographical location and communication, these clouds were not able to achieve optimal energy savings. There is a need for more work to research the use of federated clouds by leveraging the differences in energy sources resulting from geographical variations and achieve significant energy savings. In this study, we addressed the previous research gap in carbon emissions and also considered the energy consumption from cooling systems.

Methodology

Problem definition

The problem of scheduling DAG tasks in federated cloud environments is often described as a mapping problem. By mapping a given set of tasks to a given set of resources in an optimal way, the utilization of resources is improved while accelerating task execution. This federated cloud system (FC) consists of multiple cloud service providers' data centers (DCs), each cloud data center contains multiple machine rooms (MHs), and each machine room contains multiple servers (PMs). We describe the federated cloud system as:

$$FC = \{DS_1, DS_2, ..., DS_n\} \tag{1}$$

where $\boldsymbol{DS_i}(\boldsymbol{i} = 1, 2, ..., \boldsymbol{n})$ denotes the data center of cloud service provider $\boldsymbol{i}$. Datacenter is represented as:

$$DS_i = \{MH_1, \mathrm{MH}_2, ...MH_i, ...MH_n\} \tag{2}$$

where DS_i and $\mathrm{MH}_j(j = 1, 2, ..., n)$ represents room j in data center i. Machine room is represented as:

$$\mathrm{MH}_k = \{server_1, server_2, ..., server_i, ..., server_n\} \tag{3}$$

where MH_k and $\mathrm{server}_i(i = 1, 2, ..., i, ..., n)$ denotes the i-th server in the k-th server room.

A DAG task (DT) is usually composed of multiple subtasks, which are accompanied by a large number of constraint relationships between subtasks. The disassembly of a DAG task, the dependency of execution time between subtasks, and the way of communicating between the tasks have a great impact on the task's efficiency of execution and the utilization of the resources. This relationship can be described as:

$$\mathrm{DT} = \{task_1, task_2, \ldots, task_i, \ldots task_\partial\} \tag{4}$$

where $\mathrm{task}_i(i = 1, 2, \ldots, \partial)$ denotes the i-th subtask in the DT and ∂ denotes the number of subtasks. There is a dependency between subtasks in execution can be described as:

$$\mathrm{task}_i = \{st, pt, pre\_task\} \tag{5}$$

where *st*, *pt* and *pre_task* denote the start time of the task, the execution time of the task, and the set of other subtasks on which the execution of the task depends, respectively. $\mathrm{pre}\_task_i = \{task_j, \ldots, task_k\}$ indicates that the execution of $\mathrm{task_i}$ needs to depend on multiple subtasks in the precedence sequence.

During the disassembly of tasks, the main consideration is the way of data transfer between different tasks, as shown in Fig. 1. The dependency between tasks can be mapped to the start time of a subtask. The delay time of the current subtask compared to the start of the execution of the parent task can be described as

$$\mathrm{lt} = st - min(st_j, ..., st_k) \tag{6}$$

In the task scheduling process, the primary focus is on minimizing task latency through using Advantage Actor Critic (A2C) to decide the allocation of tasks to specific data center rooms. For mapping of subtasks and servers within the DAG, the HEFT algorithm is used.

Reinforcement learning environment modeling

The scheduling for real-time tasks in a federated cloud environment is essentially a continuous decision process, and the use of RL can transform this scheduling problem

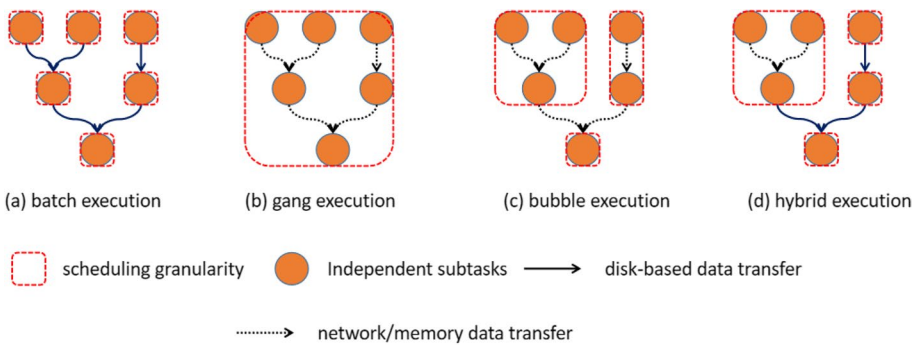


Fig. 1 DAG disassembly methods

into a Markov or Semi-Markov decision problem. Therefore, this paper models the scheduling problem as a triplet (*S, A, R*), where *S* denoting the state, *A* representing the action, and *R* representing the reward.

Characteristic expression of the state

The state describes the characteristics of the scheduling environment, which contains both local and global characteristics. In RL, the actions made by the agent to interact with the environment depend entirely on the state of the environment. The state is modeled as S ($minWT_i$, $maxWT_i$, TE_i, CT_i, *TEN, TEM, ET, ER, CNT, CMET*).

Where $minWT_i$ and $maxWT_i$ denote the shortest wait time and the longest wait time of the No. i room, which is constructed as:

$$minWT_i = min(met_1, met_2, ..., met_i, ..., met_n) \quad (7)$$

$$maxWT_i = max(met_1, met_2, ..., met_i, ..., met_n) \quad (8)$$

Where met_i denotes the remaining execution time of machine i. It is assumed that each machine can execute only one independent subtask at a given time.

Energy consumption in a data center is mainly composed of two components: one is the energy consumption generated by IT equipment and the other is the energy consumption generated by infrastructure. The energy consumption of IT equipment is mainly generated by servers, the energy consumption of infrastructure is mainly generated by cooling equipment. There are two main cooling methods used in data centers: air cooling and liquid cooling, where the main air-cooling energy consumption is generated by the blower, and its power consumption is typically around one-third of the blower speed. According to the heat transfer theory and the general fan law, the energy consumption generated by the blower and the energy consumption generated by the server can reach a fixed ratio under strict control of the blower speed [39]. The energy consumption generated by these two cooling methods is different, and in order to better reduce the cooling energy consumption, the different cooling methods can be dynamically adjusted according to the current load of each server room in the data center. Thus, the energy consumption of a single server room in a data center can be modeled as:

$$TE_i = \sum_{k=1}^{n} (exeTime_k * P_{peak} + idleTime_k * P_{idle}) * (1 + cr) + P_{ctStart} \quad (9)$$

TE_i is the total energy consumption of server room i, $exeTime_k$ is the duration of server k in the computing state, and P_{peak} is the peak power of the server. $idleTime_k$ is the duration of server k in the standby state, where *idleTime* is composed of two parts of the standby time: (1) the waiting time for the child tasks on the current server to wait for the completion of the execution of the parent task on which it depends and (2) the waiting time for all the remaining subtasks to be completed. P_{idle} represents the power of the server in standby state. *cr* represents the ratio between the server energy consumption and the cooling energy consumption. $P_{ctStart}$ represents the startup energy consumption due to the switching of different cooling methods and can be modeled as:

$$P_{ctStart} = \begin{cases} P_{airStart} & exe(server) * P_{peak} < P_{airPeak} \& CT_{current} = Liquid \\ P_{liquidStart} & exe(server) * P_{peak} > P_{airPeak} \& CT_{current} = Air \\ 0 & other \end{cases} \quad (10)$$

$P_{airStart}$ is the start-up power for air cooling, $P_{liquidStart}$ is the start-up power for liquid cooling, *exe*(*server*) is the number of servers in the current server room in the computing state, $P_{airPeak}$ is the peak cooling power for air cooling, and CT_i is the cooling method for server room i in the current data center.

The above states reflect local characteristics of the current data center including the load balance, energy consumption and cooling methods of each room in the current data center. The following will describe the data center in terms of global characteristics. *TEN* and *TEM* represent the total energy consumption and emissions of the current data center, respectively. *ET* represents the type of energy used in the current data center, which is the key to the federated cloud ability to reduce carbon emissions, because the federated cloud is composed of several geographically distributed cloud service providers. The type of energy used is different in different geographic locations, for example, most of the electricity in Norway comes from hydroelectric power, while France mainly relies mainly on nuclear energy [49]. The emission ratio of the same energy consumption is different because of the different energy types. *ER* represents the emission ratio of the energy used in the data center to the carbon emissions. *CNT* represents the number of subtasks contained in the current DAG, and *CMET* represents the execution time of the largest subtask in the

current DAG. All the above states are global states, which reflect the overall operation of the current data center and thus help the agent to make decisions, where *TEN* and *TEM* can be modeled as follows:

$$TEN = \sum_{i=1}^{n} TE_i \tag{11}$$

$$TEM = ER * \sum_{i=1}^{n} TE_i \tag{12}$$

where n indicates the current number of data center server rooms.

Action definition

In this paper we define the action as the choice of the most appropriate data center to allocate the incoming DAG tasks, with the aim of minimizing the energy and carbon emissions generated at the end of a batch of task scheduling. The action space is constructed as follows:

$$action = (prob_1, prob_2, ..., prob_i, ..., prob_n, prob_{nothing}) \tag{13}$$

where $prob_i$ represents the probability of assigning the current DAG task to room i and $prob_{nothing}$ represents the probability of not assigning the DAG task to the current data center. To enhance the exploration capability, the selection of actions from the action space is based on probability.

Reward function

In RL, the design of the reward is crucial, as it directly reflects the effectiveness of the current decision made by the agent. However, in practice, the feedback from the reward is often delayed. For DAG task scheduling, after the task is assigned to the data center, the correctness of the agent's decision is not immediately available since the energy consumption and emissions generated by the execution of the task are not immediately available. Therefore, the rewards can be constructed as follows:

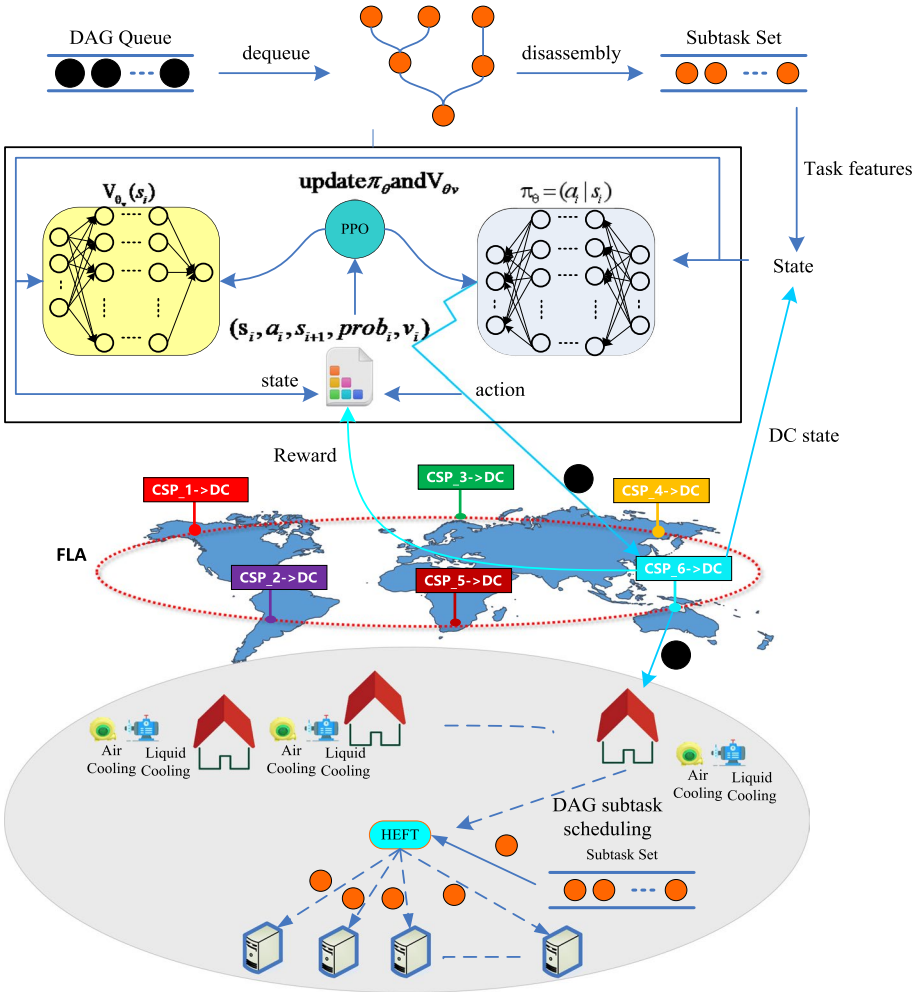


Fig. 2 The overall system architecture of ERLFC

$$reward = \begin{cases} \left(\frac{E(\mathrm{RH})\text{-}E(ERLFC)}{E(RH)} + \frac{EMI(RH)-EMI(ERLFC)}{EMI(RH)}\right) * \vartheta & done = true \\ 0 & done = false \end{cases} \tag{14}$$

where $E(RH)$ represents the total energy consumption generated by using RoundRobin and HEFT (RH) joint scheduling, $E(ERLFC)$ represents the total energy consumption generated by ERLFC scheduling policy scheduling, $EMI(RH)$ represents the total emission generated by using RH scheduling, $EMI(ERLFC)$ represents the total emission generated by ERLFC scheduling policy scheduling, and ϑ represents the reward amplification ratio, which can be amplified *reward* to help agent to update in a better direction. The sum of these two ratios ensures that when the energy consumption and emissions generated by ERLFC scheduling are smaller than those generated by RH scheduling, the model receives positive feedback and can thus reinforce the learned policy.

System architecture and algorithm design

Figure 2 shows the overall system architecture of ERLFC. The ERLFC schedules DAG tasks in a real-time environment where the arrival of DAG tasks is not time bounded. To ensure the reliability of the overall scheduling, tasks are placed into a task queue when they arrive, and the queue can be represented as $Q = \{DT_1, DT_2, \ldots DT_i, \ldots, DT_n, \}$. The detailed implementation of our proposed ERLFC architecture is accessible on GitHub [50].

The ERLFC model requires two networks in the training process: 1) DAG task scheduling policy network (actor) $\pi_\theta = (a_i|s_i)$, which calculates the probability distribution a_i of each action in the action space based on the input state s_i and the weight θ. 2) value network (critic) $V_{\theta v}(s_i)$, which calculates the value of the state based on the input state S_i and the weight θ_v. Both networks need to be optimized during the training process so that the agent can get a more optimal objective function *reward* when making the final decision.

The optimization of π_θ relies on the strategy gradient theorem, and in updating the strategy parameters θ using the strategy gradient theorem, $A(\tau)\nabla \log p(a_t|s_t, \theta)$ is used, where $A(\tau)$ is evaluated by critic, which is called the advantage function $A(\tau) = r_t - (V^{\pi_\theta}(s_t) - V^{\pi_\theta}(s_{t+1}))$. where r_t represents the real *reward* that the agent gets when it sees state S_t and takes action a_t, and $V^{\pi_\theta}(s_t) - V^{\pi_\theta}(s_{t+1})$ represents the *reward* that the agent is expected to get when it sees state s_t. When the advantage function gets the value $A(\tau) > 0$, the probability of action a_t can be increased, and vice versa, the probability of action a_t can be decreased.

The specific design of the Actor-Critic network is illustrated in Fig. 3 and the specific parameters for network architecture are provided in Table 1. In a real-time environment the *CNT* and *CMET* of the future DAG tasks are usually unbounded, and the normalization of *CNT* and *CMET* is required to be handled specially in order for the Actor to perceive the importance of the current DAG task. The normalization method we use is as in Eqs. (15) and (16):

$$\mathrm{CNT} = \frac{\partial}{max(h(\partial), \partial)} \tag{15}$$

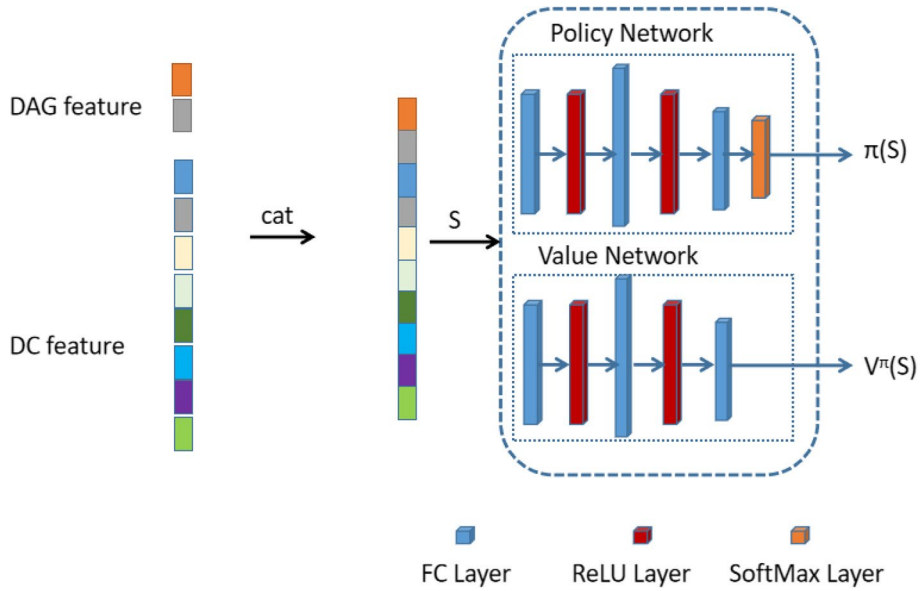


Fig. 3 Overview of ERLFC architecture

Table 1 Relevant parameters of ERLFC

Parameters	Value
Input	39
Actor FC Layer	64
Actor FC Layer	128
Actor FC Layer	8
Critic FC Layer	64
Critic FC Layer	128
Critic FC Layer	1
Optimizer	Adam
Loss function	MSELoss
Learning rate of actor	0.001
Learning rate of critic	0.01

$$\mathrm{CMET} = \frac{max(pt)}{max(h(pt), max(pt))} \tag{16}$$

The normalization method allows the model to continuously learn during the training. When the value of *CNT* or *CMET* is 1, it means that the scale or execution time of the current task is never seen in the historical tasks, and the model should place more emphasis on the task characteristics in the decision-making process.

During the agent training process, we incorporate proximal policy optimization (PPO) [51], which combines the benefit of trust region policy optimization (TRPO) [52] and allows for multi-cycle small batch updates of the agent. PPO uses the following objective function in Eq. (17) to update the policy network:

$$L(\theta) = \mathbb{E}_t[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t)] \tag{17}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta\,old}(a_t|s_t)}$, $\pi_{\theta_{old}}$ represents the policy network before θ updates, $A_t = r_t - (V^{\pi_\theta}(s_t) - V^{\pi_\theta}(s_{t+1}))$ is the advantage function, and *clip* is a clipping function that keeps $r_t(\theta)$ within the range $[1-\varepsilon, 1+\varepsilon]$, in this paper $\varepsilon = 0.2$.

The agent is responsible for assigning tasks to specific rooms in the specific data centers based on the characteristics of the DAG tasks, as well as the global state of data centers and server rooms in the federated cloud. The reason why subtasks in a DAG are not used as independent schedulers in task assignment is that data also generates a lot of energy consumption and emissions when they are transmitted over the network [13]. By assigning all tasks in a DAG to the same room, we can avoid the energy consumption caused by data transmission over the network and the computational instability caused by network delays. Additionally, we can use the overlap of data transmission and computation to ignore the time required for communication. The concept of overlapping of computation and communication has been demonstrated in [53].

In this paper, we adopt the HEFT method to assign independent subtasks to server rooms because HEFT usually assigns tasks with a global perspective and can minimize the makespan of the entire DAG task. The principle of assigning a separate subtask to a server follows the formula in Eq. (18):

$$< server_i, task_j >= \mathrm{argmin}\left(\left|(lt + ms) - mrs_i\right|\right) \tag{18}$$

where $<server_i, task_j>$ represents the assignment of subtask j to server i. ms represents the smallest remaining execution time among all servers in the current server room. mrs_i represents the status (remaining execution time) of server i in the current server room.

After assigning subtask $task_j$ to $server_i$ for completion, the update strategy for mrs_i is:

$$\mathrm{mrs}_i = \begin{cases} freeTime_i + exeTime_j & lt + ms > mrs_i \\ exeTime_j & other \end{cases} \tag{19}$$

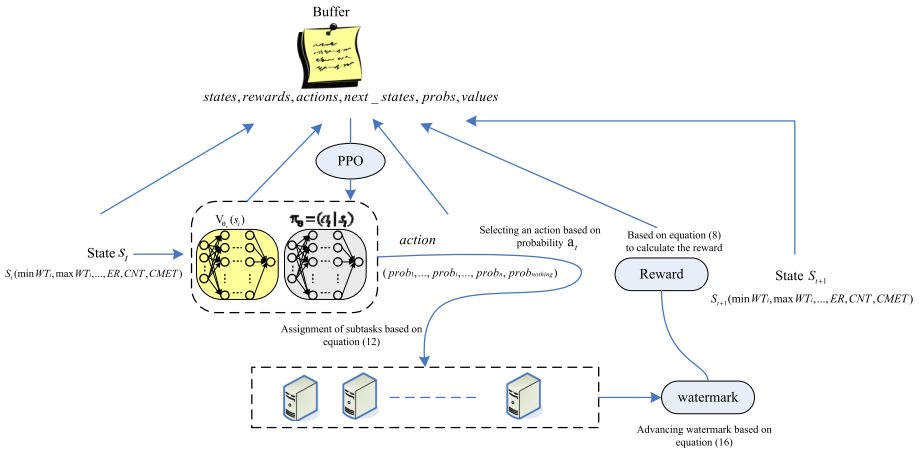


Fig. 4 Workflow of ERLFC at the t iteration for scheduling policy

where $freeTime_i$ represents the extra period of idle time that $server_i$ has before the execution of $task_j$, and $exeTime_j$ represents the execution time of $task_j$. Where $freeTime_i$ can be constructed as follows:

$$freeTime_i = (it + ms) - mrs_i \tag{20}$$

In order to better simulate the arrival of real-time tasks in real-world, we introduce the concept of watermark in our experiments [54]. The watermark represents a marker to measure the event time progress, which declares the time t at which the event time has been reached in the simulated environment. In the experiments watermark is constructed as shown in Eq. (21):

$$watermark = round(min(RET^1_{1_1}, RET^1_{1_2}, ..., RET^n_{k_i})) * (1 + wr) \tag{21}$$

where $RET^n_{k_i}$ represents the remaining execution time of server i in server room k in data center n and wr represents the watermark advance rate. The main goal is to have an indeterminate number of tasks finish executing in the entire federated cloud after completing one watermark advance, rather than just one.

Table 2 Data center configuration parameters

Parameters	Value
Static energy consumption of a single server	0.15kw/h
Peak energy consumption of a single server	0.5kw/h
Calculation of energy consumption and air cooling energy consumption ratio	0.0005
Calculation of energy consumption and liquid cooling energy consumption ratio	0.2
Peak power for air cooling	1.55kw/h
Air-cooled starting power	0.525kw
Starting power for liquid cooling	1.5kw
Fire energy emission ratio	0.6
Solar emission ratio	0.3
Wind energy emission ratio	0.1
Water to energy discharge ratio	0.3
Hydrogen energy emission ratio	0.2
Nuclear energy emission ratio	0.3

Table 3 Details of the dataset

Dataset No	Number of DAG tasks	Number of subtasks
1	1,633	43,940
2	5,003	149,176
3	15,195	472,658

Algorithm 1 Training agent for ERFLC

Input: DAG queue and data center

Output: The policy network π_θ

1: Create an $states, rewards, actions, next\_states, probs, values$ list to record the states, rewards, actions and probabilities of the actions generated by the agent in this epoch decision and the values predicted by the value network seeing the states

2: Reset data center

3: **for** task **in** DAG queue **do**

4: Store s_t state to $states$

5: $\pi_\theta(a_t|s_t)$ calculates $action$ based on state s_t as in Equation (13), $V_{\theta_v}(s_i)$ calculates the value of state s_t and stores it to $values$

6: Choose to assign the task to the server room of the current data center according to the probability, or continue to explore the next data center, and return to 5 if exploring the next data center, otherwise record the action and the probability of the action to $actions$ and $probs$

7: The set of subtasks is sent to the corresponding data center and each subtask in the set of subtasks is assigned to which server in the server room for execution according to Equation (18)

8: Advancing watermark based on Equation (21)

9: Energy consumption and emissions of computer rooms and data centers based on Equations (9), (11) and (12)

10: Calculate the reward generated by this task scheduling based on Equation (14)

11: Split the next DAG task into multiple independent subtasks and store them in the subtask collection

12: Extract data center state minWT_i, maxWT_i, TE_i, CT_i, TEN, TEM, ET, ER and normalize to the maximum value

13: Extract task features CNT and $CMET$ and normalize using Equations (15) and (16)

14: Concatenate data center state and task characteristics as the next state s_{t+1} of the data center and store s_{t+1} to $next\_states$

15: **end for**

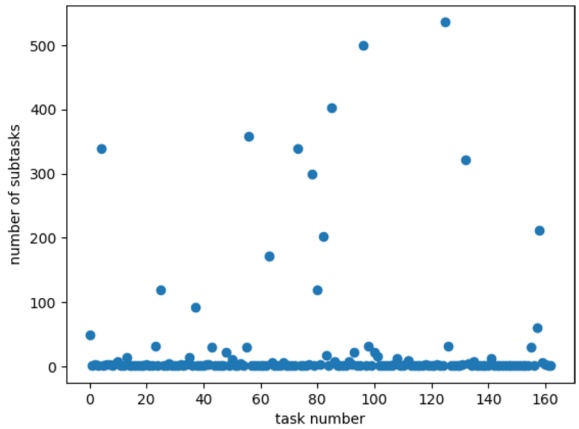
16: **for** num **in** range(100) **do**

17: Use $states, rewards, actions, next\_states, probs, values$ and depend on Equation (17) to update the parameters θ and θ_v of actor and critic

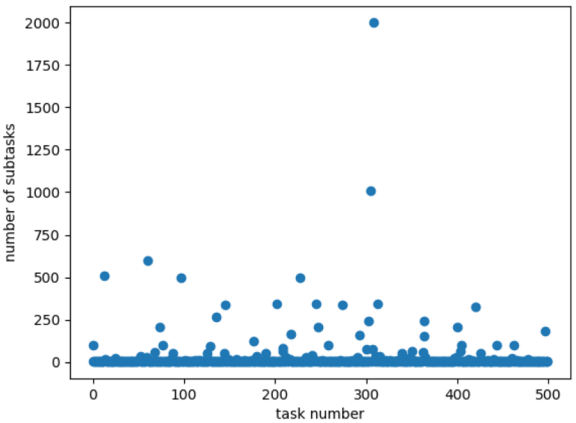
18: **end for**

19: clear $states, rewards, actions, next\_states, probs, values$

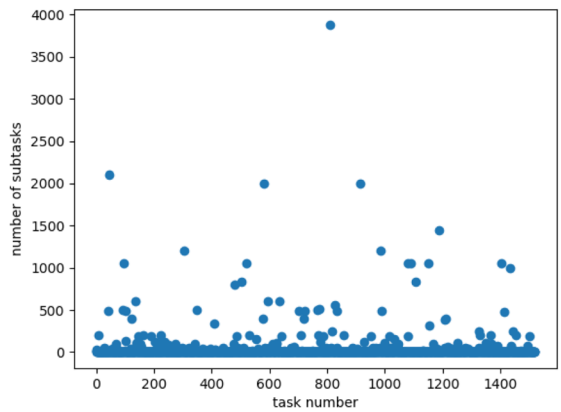
Algorithm 1. Training process of agent



(a) Distribution of sampling tasks for dataset 1

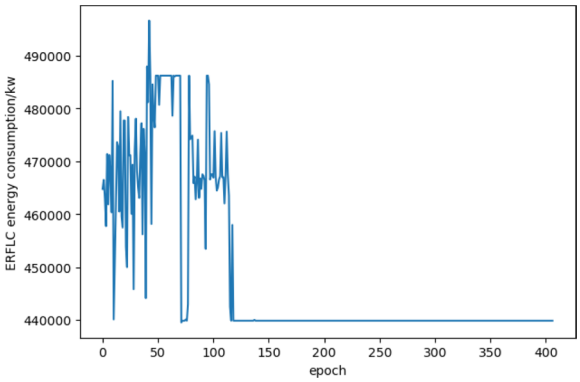


(b) Distribution of sampling tasks for dataset 2

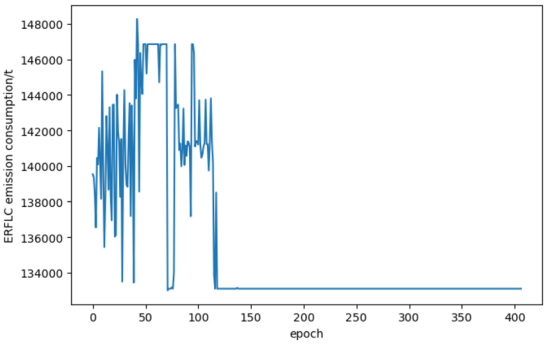


(c) Distribution of sampling tasks for dataset 3

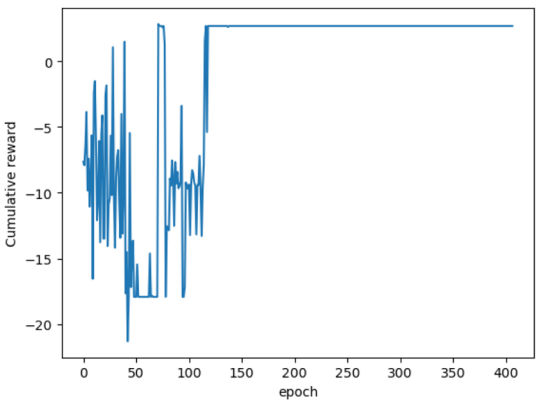
Fig. 5 Distribution of dataset sampling tasks



(a) Energy consumption training curve



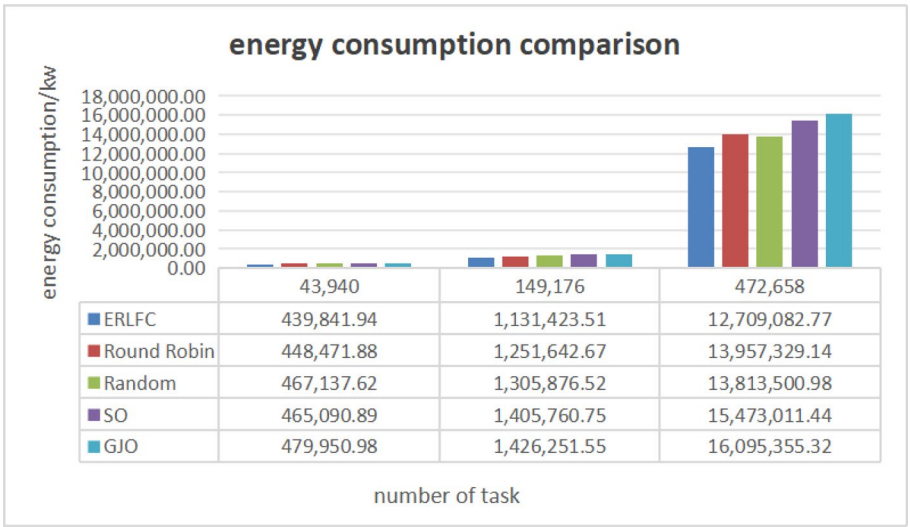
(b) Carbon emission training curve



(c) Rewards training curve

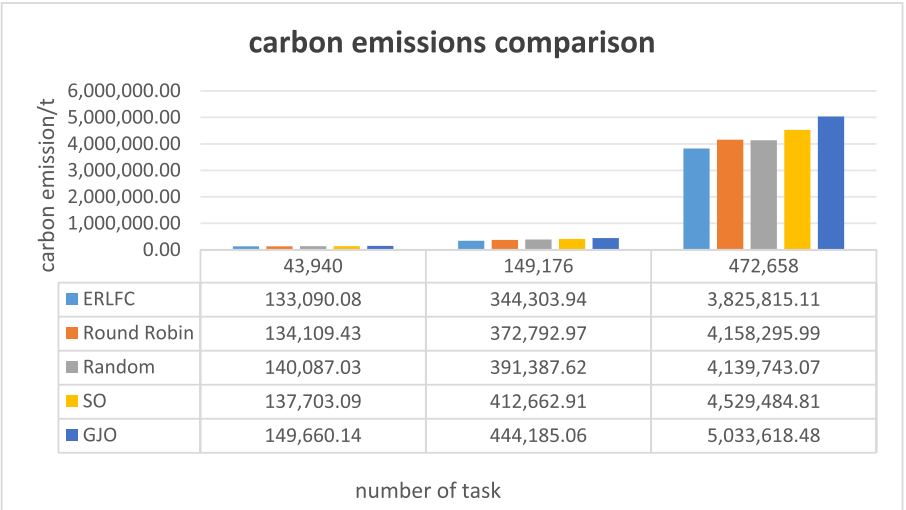
Fig. 6 Training curves for ERLFC training process

Algorithm 1 gives the training process of the agent in the kth epoch of the ERLFC model. Before the training, the state of the data center is initialized, and a set is created to record the parameters of the training process (lines 1–2). During the training process, the task is first assigned to the server room of the data center for execution using the policy network $\pi_\theta(a_t|s_t)$ based on the state S_t (lines 5–6). Then HEFT is used to assign the subtasks to the corresponding servers for execution, and the watermark is advanced to end part of the task execution. The energy consumption and emission generated by this watermark advancing to the server rooms of each data center and the *reward* generated by this scheduling can be calculated according to the advancing watermark (lines 7–10). After the watermark advance is completed, the next state of the data center and the characteristics of the next task to be scheduled are calculated and spliced into a new state S_{t+1}, which serves as the basis for the agent's decision (lines 11–14). Once the task scheduling is completed, the PPO algorithm is used to update the policy and value networks based on the historical decision information (16–18). Figure 4 shows the workflow of ERLFC in the t iteration.



	43,940	149,176	472,658
ERLFC	439,841.94	1,131,423.51	12,709,082.77
Round Robin	448,471.88	1,251,642.67	13,957,329.14
Random	467,137.62	1,305,876.52	13,813,500.98
SO	465,090.89	1,405,760.75	15,473,011.44
GJO	479,950.98	1,426,251.55	16,095,355.32

(a) Comparison of energy consumption



	43,940	149,176	472,658
ERLFC	133,090.08	344,303.94	3,825,815.11
Round Robin	134,109.43	372,792.97	4,158,295.99
Random	140,087.03	391,387.62	4,139,743.07
SO	137,703.09	412,662.91	4,529,484.81
GJO	149,660.14	444,185.06	5,033,618.48

(b). Comparison of carbon emissions

Fig. 7 Evaluation results with different scheduling algorithms

Experiments and results

Experimental environment setup

The federated cloud we consider in this study is composed of six data centers with geographically dispersed CSPs. Each data center has a data structure similar to that described in [39], consisting of seven server rooms, with each room containing 10 servers. Consequently, a single data center has 70 servers, and the entire federated cloud contains 420 servers, each sharing identical server configurations across all data centers. We introduce six types of energy sources, one corresponding to one of the six cloud data centers: (cloud data center 1, fire energy), (cloud data center 2, solar energy), (cloud data center 3, wind energy), (cloud data center 4, hydro energy), (cloud data center 5, hydrogen energy), and (cloud data center 6, nuclear energy). Additional configurations are shown in Table 2.

In this paper, we conducted experiments using real trace data from the Google Cluster dataset [55] to investigate the impact of ERLFC on energy consumption and carbon emissions in a federated cloud environment. Each task in the dataset is represented by a DAG, and each DAG subtask is as in Eq. (5). The task execution time is estimated during task scheduling and can be predicted based on historical execution data [56]. In order to evaluate the stability of the model's decisions in a real-time environment, we tested the model using three different numbers of tasks, and the details of the datasets are shown in Table 3. To analyze the distribution of the subtask numbers in the DAG tasks in the different datasets, we randomly sampled 1% of the data and plotted the results in Fig. 5. The results show that there are more small-scale DAGs in the datasets, but the number of subtasks in each DAGs varies widely, and the distribution is irregular, thus testing the ERLFC model's ability to make decisions in the face of unknown tasks.

Training process

The training process utilizes dataset 1, and Fig. 6 presents the energy consumption, carbon emission and cumulative reward change curves during the training. It is observed that the convergence of the agent is very fast during the training process. The PPO algorithm used in this study allows for complete exploration of each state, contributing to this rapid convergence.

Results and discussions

To evaluate the performance of ERLFC for task scheduling, we compared it with four baseline algorithms:

Round Robin algorithm: A round-robin approach is used to select data centers, and then machine rooms are selected in a round-robin fashion for task assignment.
Random algorithm: The scheduling of this task will select a random server room from a random data center.
SO algorithm [57]: Scheduling of tasks using SO algorithms taking into account of energy consumption, carbon emissions and makespan metrics.
GJO algorithm [58]: Scheduling of tasks using GJO algorithms taking into account of energy consumption, carbon emissions and makespan metrics.

In order to evaluate the effectiveness of the trained ERLFC, we use it in conjunction with the four aforementioned algorithms for scheduling varying quantities of DAG tasks and sub-tasks, and the experimental results are shown in Fig. 7.

Figure 7 shows the comparison between ERLFC and the four algorithms regarding energy consumption and carbon emission for different datasets. It can be seen that for dataset 1, RoundRobin, Random, SO and GJO consume 1.01, 1.06, 1.05 and 1.09 times more energy

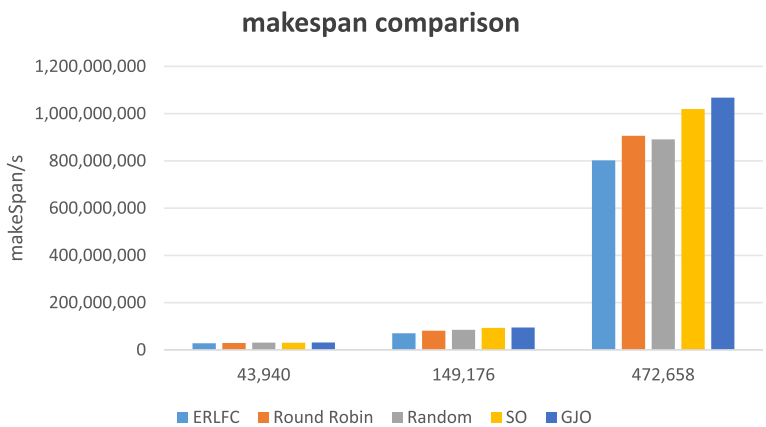


Fig. 8 Comparison of makespan

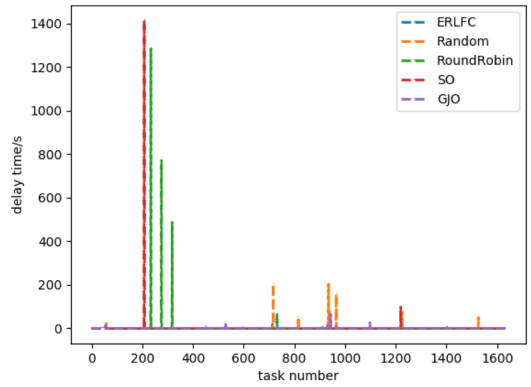
and emit 1.02, 1.06, 1.06, 1.09 times more carbon compared to ERLFC, respectively. For the dataset 2 energy consumption and carbon emissions are 1.10, 1.15, 1.24 and 1.26 times higher than those of ERLFC. For dataset 3, energy consumption and carbon emissions are 1.09, 1.08, 1.21 and 1.26 times higher than those of ERLFC. To ensure stability of the experimental results we have taken the average of the outcomes over 10 iterations for the Random algorithm.

From the experimental results, it can be seen that the heuristic scheduling algorithm (SO and GJO algorithms) can obtain better results than the traditional scheduling algorithm (RoundRobin and Random algorithms) for small data volume, mainly because the difference in the state of the data centers is relatively small, and the heuristic algorithm can obtain better results through the optimization search, and the traditional scheduling algorithm can make the load between the data centers more balanced for large data volume, but the heuristic algorithm cannot use the state information of the data centers, which leads to a significant increase in energy consumption and emissions. The heuristic algorithms are unable to use the state information of the data center which leads to a significant increase in energy consumption and emissions. ERLFC algorithm achieves better results because it uses the state information of the data center and can sense the state of the data center in the real-time environment, so that the tasks can be assigned to a more appropriate machine to execute, thus reducing energy consumption and carbon emissions. For real-time task scheduling makespan is also a more important metric and Fig. 8 shows the comparison between ERLFC and the baseline algorithms in terms of makespan.

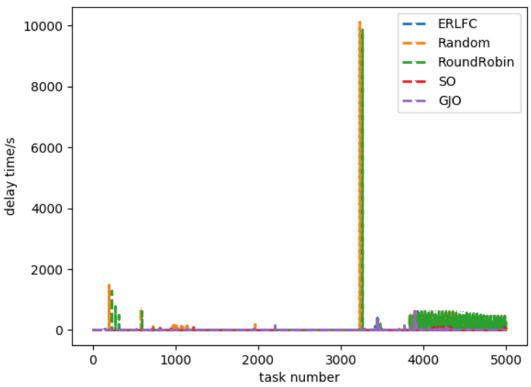
Figure 8 shows that ERLFC's reduction in energy consumption and carbon emissions does not come at the expense of makespan, and thus ERLFC's scheduling of real-time tasks is better able to satisfy service level agreements.

The task's delay is also an important indicator for assessing the scheduling performance of the model. The introduction of watermark becomes significant due to the uncertainty associated with the task's arrival time. In this study, the task delay time is primarily defined as the duration between the assignment of the task to the server and the actual execution time of the task. Considering the substantial variation in task execution times within the Google dataset, there is a heightened demand for the model's load balancing capability. Figure 9 illustrates the delay times incurred by various task scheduling algorithms during the scheduling process.

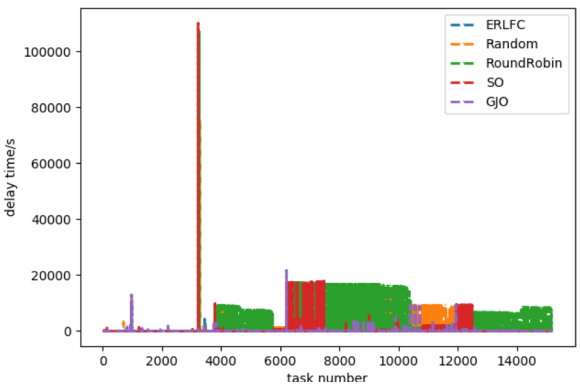
In Fig. 9, it is evident that with a small number of tasks, the delay time is shorter. However, as the number of tasks increases, a noticeable increase in task wait times is observed. Notably, the ERLFC model exhibits frequent task waiting as the task count rises on dataset 3. Despite this, in comparison to other algorithms, the overall waiting time for tasks remains relatively small. This is primarily attributed to the increment in task volume and, secondarily, to the model strategically assigning tasks for



(a) Delay time for each task in dataset1



(b) Delay time for each task in dataset2



(c) Delay time for each task in dataset3

Fig. 9 Comparison of delay time

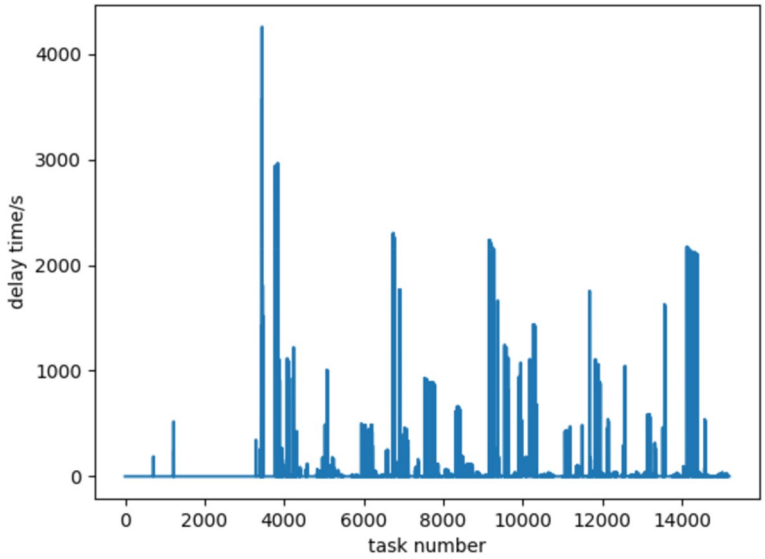


Fig. 10 Delay caused by ERLFC scheduling for dataset 3

execution in data centers with lower energy consumption and emissions. The goal is to mitigate energy usage and emissions. Given the comparatively minimal delay introduced by ERLFC in scheduling dataset 3, Fig. 10 provides a dedicated visualization of task delay.

Conclusion

The significant energy consumption and carbon emission in cloud environments have hindered the development of cloud computing. In this paper, we explore the use of reinforcement learning for real-time task scheduling on a federated cloud, taking into account the energy consumption caused by different cooling methods in data centers. In order to reduce the energy consumption and carbon emission of data centers, we use the different CSP data center locations and energy sources in the federated cloud to optimize the energy consumption and carbon emission. Additionally, we introduce Watermark to simulate the randomness of task arrival in a realistic environment. The simulation results demonstrate that the ERLFC algorithm effectively reduces data center energy consumption and carbon emissions by 1.09, 1.08, 1.21, and 1.26 times in terms of energy savings and emissions reduction when compared to the RoundRobin, Random, SO, and GJO algorithms in the scheduling of real-time tasks. Therefore, the ERLFC proposed in this paper provides an effective solution to reduce the energy consumption and carbon emission generated by data centers. For future work, we will explore the scheduling capabilities of ERLFC for specialized domains, considering task scheduling and server shutdown in heterogeneous scenarios with multiple different data center resources.

Authors' contributions
Shuaijun Chen: conducted research, wrote the initial draft, and prepared all figures and tables. Zhibao Wang and Lu Bai: planned and supervised the work. Zhibao Wang, Lu Bai, Juntao Gao, Jinhua Tao, Raymond R. Bond, and Maurice D. Mulvenna: modified and reviewed the paper. All authors reviewed the results and approved the final version of the manuscript.

Funding
This work was supported in part by TUOHAI special project 2020 from Bohai Rim Energy Research Institute of Northeast Petroleum University under Grant HBHZX202002, project of Excellent and Middle-aged Scientific Research Innovation Team of Northeast Petroleum University under Grant KYCXTD201903, Heilongjiang Province Higher Education Teaching Reform Project under Grant SJGY20200125 and National Key Research and Development Program of China under Grant 2022YFC330160204.

Availability of data and materials
The datasets and code used in this study are available for download from the following repository: https://github.com/ShuaijunC/RL_BasedTaskSchedulingForEnvironmentallSustatinableFCC/tree/master/data.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 14 June 2023 Accepted: 21 November 2023
Published online: 07 December 2023

References

1. Taherkordi A, Zahid F, Verginadis Y, Horn G (2018) Future cloud systems design: challenges and research directions. IEEE Access 6:74120–74150. https://doi.org/10.1109/ACCESS.2018.2883149
2. Hazra D, Roy A, Midya S, et al (2018) Distributed task scheduling in cloud platform: a survey[C]//Smart Computing and Informatics: Proceedings

of the First International Conference on SCI 2016, Volume 1. Springer Singapore. 183–191
3. Fan Y, Tao L, Chen J (2019) Associated task scheduling based on dynamic finish time prediction for cloud computing. Proc - Int Conf Distrib Comput Syst 2019:2005–2014. https://doi.org/10.1109/ICDCS.2019.00198
4. Assis MRM, Bittencourt LF, Tolosana-Calasanz R, et al (2016) Cloud federations: requirements, properties, and architectures[M]//Developing Interoperable and Federated Cloud Architecture. IGI Global. 1–41
5. Gu Y, Wang D, Liu C (2014) DR-Cloud: Multi-Cloud based disaster recovery service. Tsinghua Sci Technol 19(1):13–23. https://doi.org/10.1109/tst.2014.6733204
6. Rochwerger B, Breitgand D, Levy E, Galis A, Nagin K, Llorente IM, Montero R, Wolfsthal Y, Elmroth E, Cáceres J, Ben-Yehuda M, Emmerich W, Galán F (2009) The Reservoir model and architecture for open federated cloud computing. IBM J Res Dev 53(4):1–17. https://doi.org/10.1147/JRD.2009.5429058
7. Yuan H, Bi J, Zhou MC (2022) Energy-efficient and QoS-optimized adaptive task scheduling and management in clouds. IEEE Trans Autom Sci Eng 19(2):1233–1244. https://doi.org/10.1109/TASE.2020.3042409
8. Luo L, Wu W, Di D, Zhang F, Yan Y, Mao Y (2012) A resource scheduling algorithm of cloud computing based on energy efficient optimization methods. 2012 Int Green Comput Conf IGCC 2012 (July 2007):0–5. https://doi.org/10.1109/IGCC.2012.6322251
9. Dinesh Reddy V, Gangadharan GR, Rao GSVRK (2019) Energy-aware virtual machine allocation and selection in cloud data centers. Soft Comput 23(6):1917–1932. https://doi.org/10.1007/s00500-017-2905-z
10. Kurp P (2008) Green computing. Commun ACM 51(10):11–13. https://doi.org/10.1145/1400181.1400186
11. Wang D (2008) Meeting green computing challenges. 10th Electron Packag Technol Conf EPTC 2008(858):121–126. https://doi.org/10.1109/EPTC.2008.4763421
12. Zhao X, Ma X, Chen B, Shang Y, Song M (2022) Challenges toward carbon neutrality in China: strategies and countermeasures. Resour Conserv Recycl 176(October 2021):105959. https://doi.org/10.1016/j.resconrec.2021.105959
13. Aldossary M, Alharbi HA (2022) An eco-friendly approach for reducing carbon emissions in cloud data centers. Comput Mater Contin 72(2):3175–3193. https://doi.org/10.32604/cmc.2022.026041
14. Mata-Toledo R, Gupta P (2010) Green data center: how green can we perform? J Technol Res 2:1–8
15. Forrest W, Kaplan JM, Kindler N (2008) Data centers: how to cut carbon emissions and costs[J]. McKinsey Bus Technol 14(6):4–13
16. Andrae A, Edler T (2015) On global electricity usage of communication technology: trends to 2030. Challenges 6(1):117–157. https://doi.org/10.3390/challe6010117
17. Assis MRM, Bittencourt LF (2016) A survey on cloud federation architectures: identifying functional and non-functional properties. J Netw Comput Appl 72:51–71. https://doi.org/10.1016/j.jnca.2016.06.014
18. Moreno-Vozmediano R, Huedo E, Llorente IM, Montero RS, Massonet P, Villari M, Merlino G, Celesti A, Levin A, Schour L, Vázquez C, Melis J, Spahr S, Whigham D (2016) BEACON: A cloud network federation framework. Commun Comput Inf Sci 567(644048):325–337. https://doi.org/10.1007/978-3-319-33313-7_25
19. Celesti A, Tusa F, Villari M (2012) Toward cloud federation: concepts and challenges[M]//Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice. IGI Global. 1–17
20. Dayarathna M, Wen Y, Fan R (2016) Data center energy consumption modeling: a survey. IEEE Commun Surv Tutorials 18(1):732–794. https://doi.org/10.1109/COMST.2015.2481183
21. Nor NM, Hussin M, Abdullah R (2019) Energy-saving framework for data center from reduce, reuse and recycle perspectives. Pertanika J Sci Technol 27(3):1259–1277
22. Wan J, Gui X, Zhang R, Fu L (2018) Joint cooling and server control in data centers: A cross-layer framework for holistic energy minimization. IEEE Syst J 12(3):24–2472. https://doi.org/10.1109/JSYST.2017.2700863
23. Ohadi MM, Dessiatoun SV, Choo K, Pecht M, Lawler JV (2012) A comparison analysis of air, liquid, and two-phase cooling of data centers. Annu IEEE Semicond Therm Meas Manag Symp 58–63. https://doi.org/10.1109/STHERM.2012.6188826
24. Habibi Khalaj A, Halgamuge SK (2017) A Review on efficient thermal management of air- and liquid-cooled data centers: from chip to the cooling system. Appl Energy 205(August):1165–1188. https://doi.org/10.1016/j.apenergy.2017.08.037
25. Aldossary M, Alharbi HA (2021) Towards a green approach for minimizing carbon emissions in fog-cloud architecture. IEEE Access 9:131720–131732. https://doi.org/10.1109/ACCESS.2021.3114514
26. Topcuoglu H, Hariri S, Society IC (2002) Performance-Effective and Low-Complexity. 13(3):260–274
27. Hogade N, Pasricha S, Siegel HJ, Maclejewski AA, Oxley MA, Jonardi E (2018) Minimizing energy costs for geographically distributed heterogeneous data centers. IEEE Trans Sustain Comput 3(4):318–331. https://doi.org/10.1109/TSUSC.2018.2822674
28. Ben Alla H, Ben Alla S, Touhafi A, Ezzati A (2018) Deadline and Energy Aware Task Scheduling in Cloud Computing. 2018 4th Int Conf Cloud Comput Technol Appl Cloudtech 2018. https://doi.org/10.1109/CloudTech.2018.8713338
29. Xu X, Cao L, Wang X (2016) Resource pre-allocation algorithms for low-energy task scheduling of cloud computing. J Syst Eng Electron 27(2):457–469. https://doi.org/10.1109/JSEE.2016.00047
30. Xie G, Zeng G, Xiao X, Li R, Li K (2017) Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems. IEEE Trans Parallel Distrib Syst 28(12):3426–3442. https://doi.org/10.1109/TPDS.2017.2730876
31. Safari M, Khorsand R (2018) Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment. Simul Model Pract Theory 87(July):311–326. https://doi.org/10.1016/j.simpat.2018.07.006
32. Tang Z, Qi L, Cheng Z, Li K, Khan SU, Li K (2016) An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. J Grid Comput 14(1):55–74. https://doi.org/10.1007/s10723-015-9334-y
33. Baskiyar S, Abdel-Kader R (2010) Energy aware DAG scheduling on heterogeneous systems. Cluster Comput 13(4):373–383. https://doi.org/10.1007/s10586-009-0119-6
34. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
35. Peng H, Wen WS, Tseng ML, Li LL (2019) Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment. Appl Soft Comput J 80(2019):534–545. https://doi.org/10.1016/j.asoc.2019.04.027
36. Kessaci Y, Melab N, Talbi EG (2013) A Pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation. Cluster Comput 16(3):451–468. https://doi.org/10.1007/s10586-012-0210-2
37. Ahmad F, Vijaykumar TN (2010) Joint optimization of idle and cooling power in data centers while maintaining response time. ACM SIGPLAN Not 45(3):243–256. https://doi.org/10.1145/1735971.1736048
38. Wang Y, Zhang F, Wang R, Shi Y, Guo H, Liu Z (2017) Real-time task scheduling for joint energy efficiency optimization in data centers. Proc - IEEE Symp Comput Commun 0:838–843. https://doi.org/10.1109/ISCC.2017.8024631
39. Ji K, Chi C, Marahatta A, Zhang F, Liu Z (2020) Energy Efficient Scheduling Based on Marginal Cost and Task Grouping in Data Centers. e-Energy 2020 - Proc 11th ACM Int Conf Futur Energy Syst 482–488. https://doi.org/10.1145/3396851.3402657
40. Jiang Q, Leung VCM, Tang H, Xi HS (2019) Adaptive scheduling of stochastic task sequence for energy-efficient mobile cloud computing. IEEE Syst J 13(3):3022–3025. https://doi.org/10.1109/JSYST.2019.2922436
41. Chase JS, Anderson DC, Thakar PN, Vahdat AM, Doyle RP (2001) Managing energy and server resources in hosting centers. 103. https://doi.org/10.1145/502043.502045
42. Zhang J, Cheng L, Liu C, Zhao Z, Mao Y (2023) Cost-aware scheduling systems for real-time workflows in cloud: an approach based on Genetic Algorithm and Deep Reinforcement Learning. Expert Syst Appl 234(July):120972
43. Cheng L, Wang Y, Cheng F, Liu C, Zhao Z, Wang Y (2023) A Deep Reinforcement Learning-Based Preemptive Approach for Cost-Aware Cloud Job Scheduling. IEEE Trans Sustain Comput PP 1–12. https://doi.org/10.1109/TSUSC.2023.3303898
44. Cheng L, Kalapgar A, Jain A, Wang Y, Qin Y, Li Y, Liu C (2022) Cost-aware real-time job scheduling for hybrid cloud using deep reinforcement learning. Neural Comput Appl 34(21):18579–18593. https://doi.org/10.1007/s00521-022-07477-x

45. Ding D, Fan X, Zhao Y, Kang K, Yin Q, Zeng J (2020) Q-learning based dynamic task scheduling for energy-efficient cloud computing. Futur Gener Comput Syst 108:361–371. https://doi.org/10.1016/j.future.2020.02.018
46. Siddesha K, Jayaramaiah GV, Singh C (2022) A novel deep reinforcement learning scheme for task scheduling in cloud computing. Cluster Comput 25(6):4171–4188. https://doi.org/10.1007/s10586-022-03630-2
47. Yan L, Liu W, Bai D (2019) Temperature and power aware server placement optimization for enterprise data center. Proc Int Conf Parallel Distrib Syst - ICPADS 2018:433–440. https://doi.org/10.1109/PADSW.2018.8644639
48. Kang KX, Ding D, Xie HM, Yin Q, Zeng J (2021) Adaptive drl-based task scheduling for energy-efficient cloud computing. IEEE Transactions on Network and Service Management
49. Gibney E (2022) How to shrink AI's ballooning carbon footprint. Nature 607(7920):648–648
50. Chen S (2023) 'RL_BasedTaskSchedulingForEnvironmentallSustatinableFCC', Available at: [https://github.com/ShuaijunC/RL_BasedTaskSchedulingForEnvironmentallSustatinableFCC/tree/master]. Accessed: 16/11/2023
51. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal Policy Optimization Algorithms. pp 1–12
52. Schulman J, Levine S, Abbeel P, et al (2015) Trust region policy optimization[C]//International conference on machine learning. PMLR. 1889–1897
53. Grinsztajn N, Beaumont O, Jeannot E, Preux P (2021) READYS: a reinforcement learning based strategy for heterogeneous dynamic scheduling. Proc - IEEE Int Conf Clust Comput ICCC 2021:70–81. https://doi.org/10.1109/Cluster48925.2021.00031
54. Akidau T, Begoli E, Chernyak S, Hueske F, Knight K, Knowles K, Mills D, Sotolongo D (2021) Watermarks in stream processing systems: Semantics and comparative analysis of apache flink and google cloud dataflow. Proc VLDB Endow 14(12):3135–3147. https://doi.org/10.14778/3476311.3476389
55. Charles Reiss, John Wilkes JH (2014) Google cluster-usage traces format schema 2014–11–17 external.pdf - Google Drive. Google Inc: 1–14 https://code.google.com/apis/storage/
56. Pham TP, Durillo JJ, Fahringer T (2020) Predicting workflow task execution time in the cloud using a two-stage machine learning approach. IEEE Trans Cloud Comput 8(1):256–268. https://doi.org/10.1109/TCC.2017.2732344
57. Hashim FA, Hussien AG (2022) Snake optimizer: a novel meta-heuristic optimization algorithm. Knowledge-Based Syst 242:108320. https://doi.org/10.1016/j.knosys.2022.108320
58. Chopra N, Mohsin Ansari M (2022) Golden jackal optimization: a novel nature-inspired optimizer for engineering applications. Expert Syst Appl 198(March):116924. https://doi.org/10.1016/j.eswa.2022.116924

Publisher's Note