Renewable Energy-Aware Big Data Analytics in Geo-Distributed Data Centers with Reinforcement Learning

Chenhan Xu, Kun Wang, *Senior Member, IEEE*, Peng Li, *Member, IEEE*, Rui Xia, Song Guo, *Senior Member, IEEE*, and Minyi Guo, *Fellow, IEEE*

Abstract—In the age of big data, companies tend to deploy their services in data centers rather than their own servers. The demands of big data analytics grow significantly, which leads to an extremely high electricity consumption at data centers. In this paper, we investigate the cost minimization problem of big data analytics on geo-distributed data centers connected to renewable energy sources with unpredictable capacity. To solve this problem, we propose a Reinforcement Learning (RL) based job scheduling algorithm by combining RL with neural network (NN). Moreover, two techniques are developed to enhance the performance of our proposal. Specifically, Random Pool Sampling (RPS) is proposed to retrain the NN via accumulated training data, and a novel Unidirectional Bridge Network (UBN) structure is designed for further enhancing the training speed by using the historical knowledge stored in the trained NN. Experiment results on real Google cluster traces and electricity price from Energy Information Administration show that our approach is able to reduce the data centers' cost significantly compared with other benchmark algorithms.

Index Terms—Big data, load balancing, reinforcement learning, data center

✦

1 INTRODUCTION

INTERNET services, such as web-mail, distance education, searching and online chatting, have gained great popularity in recent years. In order to achieve global service coverage and high availability, service providers have utilized multiple geo-distributed data centers to conduct streaming analytics on big data continuously collected from users [1], [2]. However, running such kinds of big data jobs on data centers consumes overwhelming amount of energy, which causes a serious burden to the environment and the economy. The statistics report [3] shows that the energy consumed by data

centers accounts for 1.3 percent of total energy consumption of the world in 2010.

There have been emphasizing research efforts on energy issues of geo-distributed data centers. Earlier studies focus on energy efficiency by employing various energy-saving techniques. e.g., dynamic CPU voltage adjusting [4] and resizing [5]. Recent works [6], [7], [8] start to exploit renewable energy, such as wind and solar, to power data centers so that the energy consumption from traditional power grid can be significantly reduced. However, how to use renewable energy to decrease the cost of big data analytics is still an open challenge [9], [10], [11].

In this paper, we investigate a renewable energy-aware job scheduling issue in geo-distributed data centers based on streaming big data analytics. Particularly, we take a set of steaming big data jobs into consideration, each of which runs on a cluster of virtual machines accommodated in several geo-distributed data centers connected to both traditional power grid and renewable energy sources with unpredictable capacity [12]. When more renewable energy has been generated at a data center due to favorable weather conditions, migrating big data jobs to this data center could decrease energy consumption from power grid under an incurred migration overhead. This cost might be high when migration meets network traffic congestion. Our proposal is designed for minimizing the total cost of energy consumption from grid and job migration without any knowledge of future renewable energy generation. To solve this challenging problem, we propose a novel job scheduling algorithm based on reinforcement learning (RL) [13] that can approximate the optimal solution by iteratively learning the feedback from historical job scheduling decisions (i.e., job locations in different

- *C. Xu is with the Key Laboratory of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210003, China. E-mail: xchank@outlook.com.*
- *K. Wang is with Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210003, China, and also with Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: kwang@njupt.edu.cn.*
- *P. Li is with the School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan. E-mail: pengli@u-aizu.ac.jp.*
- *R. Xia is with Autodesk(China)Software Research and Development Co., Ltd., Shanghai 200122, China. E-mail: rui.xia@autodesk.com.*
- *S. Guo is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China. E-mail: song.guo@polyu.edu.hk.*
- *M. Guo is with Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 201109, China. E-mail: guomy@cs.sjtu.edu.cn.*

time intervals), which are also referred to as actions. The learning process of RL consists of a sequence of actions and the corresponding rewards. In each iteration, RL maintains a value function to evaluate the expected effect of taking different actions. When the action selected by value function is applied, RL observes the state that appears thereafter, e.g., the current generated renewable energy and the data center load, the reward associated with this state that can be used to refine its value function. Although RL is a promising approach, the following challenges need to be addressed on how to quickly approximate the optimal solution for streaming big data analytics.

- *Value function design.* Designing a value function to evaluate different job scheduling actions is difficult because the system state in our problem depends on many factors, such as generated renewable energy (weather condition), resource utilization, and network congestion. Furthermore, since there is no priori knowledge of future renewable energy generation, it is difficult to evaluate the cost saving of current scheduling decision.
- *Sub-optimal and slow convergence.* The traditional RL learns knowledge from the rewards of previous scheduling decisions using temporal differences [15]. Our experiments on real data sets reveal that this method fails to guarantee quick convergence to the optimal solution that can be obtained when the knowledge of future dynamics is available.

Different from existing work striving for value function design, we propose to use neural network (NN) to evaluate different job scheduling actions that will exhibit high accuracy as shown later in our experiments. NN is composed of a number of interconnected processing elements, also called neurons, organized into several layers. Given the current state as input, it can learn how to calculate an output (i.e., expected cost) based on a set of training data. Due to the strong learning capability of NN, we integrate it into the RL framework, such that we can adjust job scheduling to approximate the optimal solution with minimum cost [16]. To further improve training accuracy, we propose a novel random pool sampling (RPS) method to improve training data quality by caching historical data and periodically retrain the NN based on them. A unidirectional bridge network (UBN) structure is designed to reuse some neurons of NN constructed in previous iterations, such that the training process can be accelerated. We summarize the contributions in this paper as follows:

- We are the first to study the streaming big data analytics with renewable energy by formulating a big data job scheduling problem in geo-distributed data centers. Our proposal is designed for minimizing the total cost of energy consumption from traditional power grid and job migration among data centers.
- We develop an RL-based algorithm to deal with the job scheduling issue, without any assumption of future renewable energy generation and job migration cost. NN is applied to simplify the value function design in the traditional RL algorithm. This paper is the first to embed NN into RL for its value function design in the studied problem.
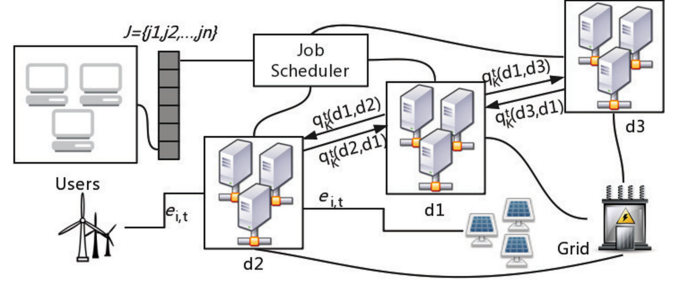

Fig. 1. System model.

- To improve the performance of our proposed scheme, two techniques are developed. In order to increase the accuracy of NN, we propose an RPS approach that periodically retrains the NN using accumulated training data with improved quality. We also design a novel unidirectional bridge network (UBN) structure to further enhance the training speed of our proposed algorithm by using historical knowledge stored in the trained NN. The experimental results show that these techniques can improve the cost saving and convergence speed by 20 and 200 percent, respectively.
- Extensive simulations are performed based on various data traces of real world workload, renewable energy, and grid electricity price for assessing the performance of our proposed algorithm. Our proposal has superior performance than benchmark approaches according to numerical results analysis.

We organize the rest of this paper as follows. Section 2 elaborates the system model and problem formulation. An RL based job scheduling algorithm is presented in Section 3. In Section 4, we enhance our proposed algorithm using a random pool sampling approach and a new NN structure. The results of extensive experiments are shown in Section 5. Section 6 surveys the related work. Section 7 gives the conclusion of our paper finally.

2 SYSTEM MODEL AND PROBLEM FORMULATION

We consider a set $D = \{d_1, d_2, \ldots d_m\}$ of data centers with renewable energy sources distributed in different geographical regions, which is shown in Fig. 1. Similar with the model adopted in [17], streaming data from users first arrive at a number of gateways, which then forward them to different data centers. Particularly, we consider a set $J = \{j_1, j_2, \ldots, j_n\}$ of big data jobs, each of which runs on a cluster of virtual machines. The data arriving rate of job j_k is denoted by c_k that would fluctuate over time. When a large amount of data arrives, the increased resource utilization of virtual clusters will lead to higher energy consumption.

Based on this observation, we consider a discrete-time power model of T time intervals, in which the total energy consumption of data center d_i at time interval t can be calculated by

$$u_{i,t} = \int_{t_0}^{t} \left[\sum_{d_i \in D} P_i(t) \right] \mathrm{d}t, \tag{1}$$

TABLE 1
Symbols and Variables

Notation	Description
D	$= \{d_1, d_2, \ldots, d_m\}$, a set of data centers
$e_{i,t}$	the renewable energy generation at data center $d_i \in D$ during time interval t
J	$= \{j_1, j_2, \ldots, j_n\}$, a set of big data jobs
c_k	the data arriving rate of job j_k
$u_{i,t}$	the energy consumption of data center d_i at time interval t
$r_i^c(t)$	the utilized CPU resource of data center d_i at time interval t
p_i^{idle}	the power of d_i when the utilization of it's CPU is at 0%
p_i^{full}	the power of d_i when the utilization of it's CPU is at 100%
$\rho_{i,t}^{\text{green}}$	the price of green energy at time interval t
$\rho_{i,t}^{\text{brown}}$	the price of brown energy at time interval t
C_{energy}^t	the energy cost at data center i in time interval t
$q_k^t(i, i')$	the cost of moving job j_k from data center i to i'
res_{j_k}	memory size of iob j_k migration
$dRatio_{j_k}$	page dirty ratio of iob j_k migration
sen_{j_k}	application sensitivity against iob j_k migration
$C_{migrate}^t$	the total job migration cost at time interval t
C	the total cost
s_t	the system state in RL
A	a set of action $a_t = \{J_1^t, J_2^t, \ldots, J_m^t\}$ in RL
r_{t+1}	the reward of action a_t in state s_t in RL
λ	the discount factor of cumulative reward in RL
$Q^\pi(s_t, a_t)$	the action-value function in RL. It calculates the action-value of a_t in state s_t
σ	the pool size in SPS or RPS
τ	the random discarding ratio in RPS
η	$\eta \in (0, 1)$ and controls the action selection in RL
M	the training epoches of RL
$F(s_t, a_t)$	the function trained by NN to approximate $Q^\pi(s_t, a_t)$
a_t^*	the action that maximizes the output of function $F(s_t, a_t)$

where

$$P_i(t) = p_i^{\text{idle}} + \left(p_i^{\text{full}} - p_i^{\text{idle}}\right)\left(2r_i^c(t) - r_i^c(t)^{1.4}\right). \quad (2)$$

In (2), $r_i^c(t)$ represents the utilized CPU resource of data center d_i at time interval t, p_i^{idle} and p_i^{full} denotes the power of d_i when the utilization of it's CPU is at 0 and 100 percent, respectively [18].

In our model, the energy supply of a data center includes green renewable energy and brown energy from grid. The policy of energy using is to give high priority to deploying the green energy, where $e_{i,t}$ is denoted as the amount of generated green energy at time interval t. $\rho_{i,t}^{\text{green}}$ and $\rho_{i,t}^{\text{brown}}$ represent the price of green and brown energy at time interval t, respectively. Energy cost at time interval t can be calculated as follows:

$$C_{energy}^t = \max\left(u_{i,t} - e_{i,t}, 0\right)\rho_{i,t}^{\text{brown}} + \min\left(u_{i,t}, e_{i,t}\right)\rho_{i,t}^{\text{green}}. \quad (3)$$

Moving a job to a different data center involves the migration of virtual clusters and redirection of data forwarding policies at gateways. Let $d(j_k, t)$ denote the data center accommodating job k in time interval t, and notation $q_k^t(i, i')$ denote the cost of moving job j_k from data center i to i'. As the result of uncertainty of performance degradation and network consumption during iob j_k migration, only core relevant factors are utilized for estimation [19]. The total job migration cost at time interval t is therefore expressed as

$$C_{migrate}^t = \sum_{j_k \in J} q_k^t(d(j_k, t), d(j_k, t-1))$$
$$= \sum_{j_k \in J} res_{j_k} * dRatio_{j_k} * sen_{j_k}, \quad (4)$$

where res_{j_k} represents memory size of iob j_k migration, $dRatio_{j_k}$ represents page dirty ratio of iob j_k migration, and sen_{j_k} represents application sensitivity against iob j_k migration.

Finally, the total cost over the whole time intervals on energy consumption and job migration can be calculated by

$$C = \sum_{t=1}^T (C_{energy}^t + C_{migrate}^t). \quad (5)$$

Our proposal is designed for minimizing the total cost by making job scheduling decisions, i.e., $\{J_1^t, J_2^t, \ldots, J_m^t\}$, at the beginning of each time interval, without the knowledge of future renewable energy generation $e_{i,t}$ and job migration cost function $q_k^t(i, i')$. The systems and variables in this paper are summarized in Table 1.

3 REINFORCEMENT LEARNING BASED LOAD BALANCING

In this section, an RL-based method is proposed for dealing with the cost minimization issue. The method overview is first presented, followed by design details.

3.1 Overview

RL is an approach to approximate the optimal reward by iteratively learning the feedback from historical decisions named actions. The learning process of RL consists of a sequence of actions and the corresponding rewards [14]. In each iteration, RL evaluates the expected effect of taking different actions by a value function. Then RL selects an action to execute, and observes the state that appears thereafter, the reward associated with this state that can be used to refine its value function. Generally, the reward is a metric of the benefit associated with an action in a certain state. Since RL do not need any priori knowledge, it is quite appropriate to the cost reduction among complex data centers.

Our proposed algorithm is mainly designed to migrate jobs among geo-distributed data centers according to the feedback of historical job scheduling decisions. We abstract our algorithm as a job scheduler, whose interaction with job execution on data centers is shown in Fig. 2a. First, the job scheduler selects an action and sends it to data centers. Then, data centers execute the action. Finally, data centers give feedback including state and reward to job scheduler for its learning. The temporal relations among these procedures is shown in Fig. 2b. In the beginning phrase of every time interval t, the job scheduler determines the locations of all jobs
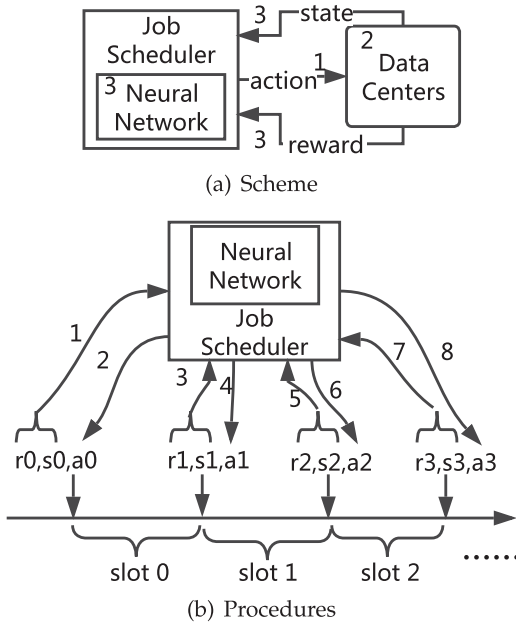
(a) Scheme

(b) Procedures

Fig. 2. The scheme and procedures of load balancing.

denoted as $a_t = \{J_1^t, J_2^t, \ldots, J_m^t\}$, which is referred to as an action. Such decision is made according to the current system state s_t that includes information as shown in Table 2. After applying action a_t, we can observe a new system state s_{t+1} associated with a reward $r_{t+1} = C_{energy}^t + C_{migrate}^t$ in the end phrase of time interval t. We model the whole system as a Markov decision process (MDP) as:

$$\{\langle s_0, a_0, r_0 \rangle, \langle s_1, a_1, r_1 \rangle, \ldots, \langle s_T, a_T, r_T \rangle\}. \quad (6)$$

In the following, we will elaborate how to determine the action a_t for each time interval t to minimize its total reward.

3.2 Algorithm Design

Given s_t and a_t, we define an action-value function $Q^\pi(s_t, a_t)$ as follows:

$$
\begin{aligned}
Q^\pi(s_t, a_t) &= \mathbb{E}\left[\frac{1}{r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \cdots} \Big| s_t, a_t \right] \\
&= \mathbb{E}\left[\frac{1}{r_{t+1} + \lambda \frac{1}{Q^\pi(s_{t+1}, a_{t+1})}} \Big| s_t, a_t \right],
\end{aligned} \quad (7)
$$

where λ is a discount factor. Let $\lambda \in [0, 1]$ so that the rewards in the nearer future have larger weights. In each time interval, we need to select the action a_t^* leading to the maximum value of function $Q^\pi(s_t, a_t)$, i.e.,

$$a_t^* = \arg\max_{a_t \in A} Q^\pi(s_t, a_t), \quad (8)$$

where A is a set of action $a_t = \{J_1^t, J_2^t, \ldots, J_m^t\}$ that can be chosen in s_t.

Since future rewards are affected by many factors, such as resource utilization of each data center and local weather conditions as shown in Table 2, the traditional reinforcement learning based on temporal differences [15] cannot accurately calculate $Q^\pi(s_t, a_t)$. Neural Network (NN) is deployed to train a function $F(s_t, a_t)$ that approximates the action-value function with high accuracy. NN can be considered as a composite function that regards state s_t as input and outputs an action a_t. As shown in Fig. 3, an example

TABLE 2
The Information Every State Contains

Attributes	Description
s_t^a	The CPU usage in current state.
s_t^b	The free RAM size in current state.
s_t^c	The free I/O bandwidth in current state.
s_t^d	The current weather.
s_t^e	The current electricity price.

neural network has 4 layers, and each layer contains a number of computing items called neurons. Each neurons receives values from all neurons in its upper layer, which are denoted by a vector x and conducts a calculation in the form of $wx + b$, where w and b are called weight and bias.

The pseudo code of our proposed RL-based algorithm is shown in Algorithm 1. We start with an initial NN $F(s_t, a_t)$ with random weight ω and bias b. In the beginning of each time interval, we generate a random number $z \in [0, 1]$ obeys uniform distribution. If $z < \eta$, we select the action a_t^* that maximizes the output of function $F(s_t, a_t)$. Otherwise, we use the randomly selected action. After applying the action a_t^*, we can observe a system state s_{t+1} at the end of time interval t, its associated reward r_{t+1} will be used to retrain the NN based on the stochastic gradient descent (SGD) method [20].

Algorithm 1. Our Proposed RL-Based Algorithm

Input: $s_0, A = \{a_0, a_1, a_2, \ldots, a_t\}, \lambda, \eta, M$
Output: (w, b)
1: **begin**
2: 　　initialize the NN $F(s_t, a_t)$ with random weight ω and bias b, $s_t = s_0$
3: 　　**for** $i \in (1, 2, \ldots, T)$ **do**
4: 　　　　**for** $j \in (1, 2, \ldots, M)$ **do**
5: 　　　　　　generate random number $z \in [0, 1]$
6: 　　　　　　**if** $z < \eta$ **then**
7: 　　　　　　　　select action $a_t^* \in A$ that maximizes the output of function $F(s_t, a_t)$
8: 　　　　　　**else**
9: 　　　　　　　　randomly select action $a_t^* \in A$
10: 　　　　　**end**
11: 　　　　　execute a_t^*, and observe reward r_{t+1} and state s_{t+1}
12: 　　　　　**retrain** (s_t, a_t^*, r_{t+1})
13: 　　　　　**begin**
14: 　　　　　　　retrain NN using (s_t, a_t^*, r_{t+1}) by SGD
15: 　　　　　**end**
16: 　　　　　set $s_t = s_{t+1}$
17: 　　　　**end**
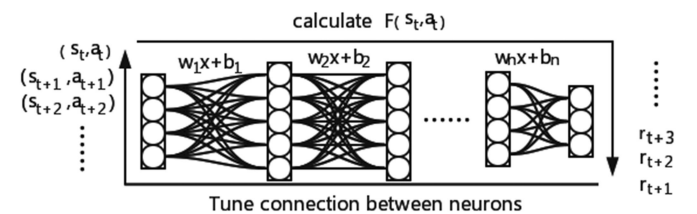18: 　　**end**
19: **end**

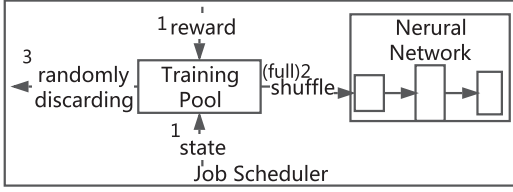---

Fig. 3. Training process.

Fig. 4. Random pool for sampling.

As shown in Fig. 3, every circle corresponds to an execution of function *retrain* in Algorithm 1. We use tuple (s_t, a_t^*, r_{t+1}) to train the NN by SGD. Algorithm calculates $wx + b$ and uses it as the input of the latter neurons. By repeating this operation, we can calculate the outputs $F(s_t, a_t^*)$ of the whole NN. According to r_{t+1}, algorithm tunes the w and b by SGD to update the network.

4 PERFORMANCE ENHANCEMENT

4.1 Random Pool Sampling

In Algorithm 1, we retrain the NN by using the SGD method that requires stochastic sampling (SS) [20] of training data. However, such requirement cannot be met because only one data record is generated and used in retraining in the end phrase of every time interval. To overcome this weakness, Simple Pool Sampling (SPS) [21] is proposed to maintaining a pool to accumulate the training data. When the pool is full, it performs training process and empties the pool. The weakness of SPS is that only the data recently come can be sampled. Alternatively, we propose a random pool sampling (RPS) scheme to approximate SS such that the accuracy of NN can be further increased. Since the RPS maintains a pool to store data, we also replace SGD with mini-batch stochastic gradient descent which generally has better accuracy than SGD [22]. As shown in Fig. 4, we maintain a pool of historical data and use them to retrain the NN based on mini-batch stochastic gradient descent. When the pool is full, we randomly discard some data records in the pool. Note that the sampling method used in Algorithm 1 is a particular case of RPS via installing the pool size to 1.

Combining RPS with mini-batch SGD, we propose a novel algorithm called *retrainOnBatch* to replace the function *retrain* in Algorithm 1. The pseudo codes are shown in Algorithm 2.

Algorithm 2. Function **retrainOnBatch** (s_t, a_t^*, r_{t+1})

1: **if** *the sampling pool is full* **then**
2:     shuffle all data in the sampling pool to generate a
        mini-batch
3:     retrain the NN $F(s_t, a_t)$ using the mini-batch SGD
4:     randomly discard $\sigma \times \tau$ data records in pool
5: **end**
6: insert the new data record (s_t, a_t^*, r_{t+1}) into the pool

---

In Algorithm 2, we first check whether the sampling pool is full or not. If it is full, we shuffle all data to generate a mini-batch that will be used by SGD to retrain the NN, as shown in lines 2 and 3. Then, we randomly discard $\sigma \times \tau$ data records, where σ is the pool size and τ is the discarding ratio. If it is not full, we just insert the new data records into the pool.

To theoretically evaluate the performance of different data sampling methods, we use the metric of Bhattacharyya distance [23] that is defined as follows.

Definition 1. *Bhattacharyya Distance is a measurement of the similarity between two discrete probability distributions. For two distributions p^* and q^* in the same region X, the Bhattacharyya distance can be written as*

$$D_B(p^*, q^*) = -ln(BC(p^*, q^*)), \qquad (9)$$

where BC is called Bhattacharyya coefficient that can be calculated by:

$$BC(p^*, q^*) = \sum_{x \in X} \sqrt{p^*(x)q^*(x)}. \qquad (10)$$

The distributions of sampling data under SS, SPS, and RPS are denoted by p_{SS}, p_{SPS} and p_{RPS}, respectively. We show the superiority of RPS to SPS, which brings further increasement of accuracy, in the following theorem.

Theorem 1. $D_B(p_{RPS}, p_{SS}) \leq D_B(p_{SPS}, p_{SS})$

Proof. The pool is assumed full at time interval t with no generality loss, and mini-batch based SGD samples σ shuffled tuples from the pool.

Since tuples are generated one by one, we denote tuples $(state_t, reward_t, action_t)$ $t \in [0, T]$ with sequence as

$$(u_0, u_1, \ldots, u_T). \qquad (11)$$

Then, we denote a sequence P as

$$(p_0^*, p_1^*, \ldots, p_T^*), \qquad (12)$$

where p_i is the probability to be sampled of the tuple u_i. The P_{SS} can be denoted as

$$P_{SS} = \left(\frac{\sigma}{T}, \frac{\sigma}{T}, \ldots, \frac{\sigma}{T} \right). \qquad (13)$$

According to the definition of SPS and RPS, P_{SPS} is formulated as

$$P_{SPS} = (0, 0, \ldots, \underbrace{1, 1, \ldots, 1}_{\sigma}, 0, \ldots, 0), \qquad (14)$$

and P_{RPS} is formulated as

$$\begin{aligned} P_{RPS} = (&\underbrace{(1-\tau)^\kappa, \ldots, (1-\tau)^\kappa}_{\sigma}, \\ &\underbrace{(1-\tau)^{\kappa-1}, \ldots, (1-\tau)^{\kappa-1}}_{\tau*\sigma}, \ldots, \\ &\underbrace{(1-\tau)^2, \ldots, (1-\tau)^2}_{\tau*\sigma}, \underbrace{1, \ldots, 1}_{\tau*\sigma}, 0, \ldots, 0), \end{aligned} \qquad (15)$$

where κ represents the times of when the pool is full, $\kappa \propto t$.

Use Eq. (10) to calculate the *Bhattacharyya Coefficient* between SPS and SS, we can get

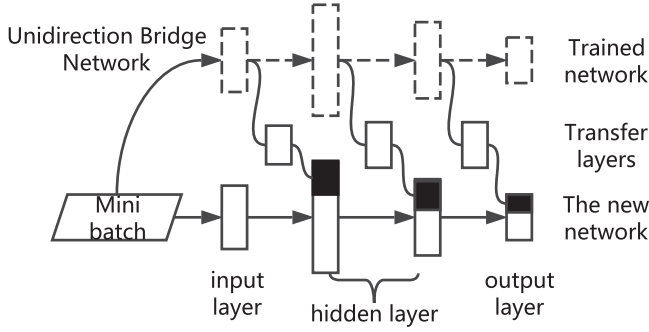$$BC(p_{SPS}, p_{SS}) = \sigma \sqrt{\frac{\sigma}{T}}. \qquad (16)$$

Fig. 5. Unidirectional bridge network structure.

Fig. 6. Details of UBN.

Similarly, the coefficient between RPS and SS is calculated by

$$
\begin{aligned}
BC(p_{RPS}, p_{SS}) =\ & \tau\sigma\sqrt{\frac{\sigma}{T}} + \tau\sigma\sqrt{\frac{\sigma}{T}}\sqrt{(1-\tau)^1} \\
& + \cdots + \tau\sigma\sqrt{\frac{\sigma}{T}}\sqrt{(1-\tau)^{\kappa-1}} \\
& + \sigma\sqrt{\frac{\sigma}{T}}\sqrt{(1-\tau)^{\kappa}} \qquad (17) \\
=\ & \sigma\sqrt{\frac{\sigma}{T}}(\tau(1+\sqrt{1-\tau}+\sqrt{(1-\tau)^2} \\
& + \cdots + \sqrt{(1-\tau)^{\kappa-1}}) + \sqrt{(1-\tau)^{\kappa}}).
\end{aligned}
$$

When the t is sufficiently large, we can get

$$
\begin{aligned}
BC(p_{RPS}, p_{SS}) =\ & \sigma\sqrt{\frac{\sigma}{T}}(1+\sqrt{1-\tau}) \\
& > \sigma\sqrt{\frac{\sigma}{T}} = BC(p_{SPS}, p_{SS}).
\end{aligned} \qquad (18)
$$

Combine Eq. (18) with Eq. (9), we can get

$$
D_B(p_{RPS}, p_{SS}) \le D_B(p_{SPS}, p_{SS}), \qquad (19)
$$

which finishes the proof. □

4.2 Unidirectional Bridge Network Structure

In this section, we further enhance our proposed RL-based algorithm by proposing a unidirectional bridge network (UBN) structure to accelerate the training process of NNs. It is motivated by the fact that neurons in the NN contain useful knowledge that can be reused in training the new NN.

In the proposed UBN structure, shown in Fig. 5, we create transfer layers between the new NN and the one trained previously. Every transfer layer contains a number of neurons that randomly collect the outputs from the trained NN. The outputs of transfer layers are sent to the new NN. Every hidden layer and output layer in the new NN receives data from both the previous layer and a transfer layer.

Since the transfer layers bring extra neurons and associated connections, we propose the random dropout [24] strategy to reduce the number of retrained neurons. As shown in Fig. 6, the dropout strategy is to randomly remove some neurons and their connections. This transfer layer can be considered as a layer with two neurons in training. Note that the dropout strategy runs at the beginning of every iteration of training, and the dropped neurons just do not
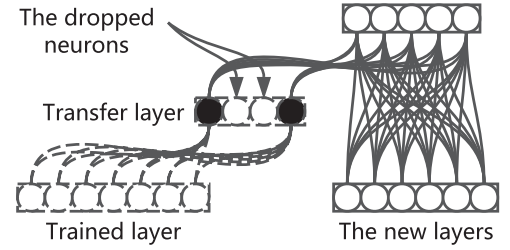
participate in the current training iteration. In every iteration of training process, the number of dropped neurons is kept at a stable value, but the neurons are uncertain.

5 EXPERIMENTS

To study the benefits of our proposed RL-based algorithm, extensive experiments based on real world data sets are performed in this section.

5.1 Experimental Setup

In our experiments, we use the following real-world data sets including data center location, workload, renewable energy generation, and electricity price of traditional power grid.

(1) Workload: We use the Google cluster-usage traces [25] collected on May, 2011, which contain 29-day job information on a cluster of about 12.5k machines.

(2) Renewable energy generation: We consider three data centers located in Prewitt in New Mexico (NM), Phoenix in Arizona (AZ) and Los Angeles in California (CA), where renewable energy is generated according to weather conditions published by National Renewable Energy Laboratory [26]. We use the energy cost and renewables for the same time. The renewable energy data includes the all-day generation of solar, water, and wind.

(3) Electricity price: The electricity price is obtained from the website of Energy Information Administration [3].

To study the influence of NN on the proposed RL-based algorithm, the discount factor λ of cumulative reward in RL is set as 0.1, 0.3, 0.5, 0.7, 0.9, and the η, which controls the action selection in RL, is set as 0.1, 0.3, 0.5, 0.7, 0.9. we consider four NN structures with different combinations of type, number of layers, and dropout polices in our experiments. As shown in Table 3, one structure uses Convolutional Neural Network (CNN) [27] whose kernel size is set to 3, and others adopt multilayer perception (MLP) [24] with one kernel. The three structures using MLP have different number of layers and dropout policies. For

TABLE 3
Different Network Structures

Structure	Type	no. of layers	dropout	kernel
A	CNN	13	Yes	3
B	MLP	13	Yes	1
C		9	Yes	1
D		5	No	1

(a) Cost

(b) Reward

Fig. 7. Cost and reward in different NN structure.
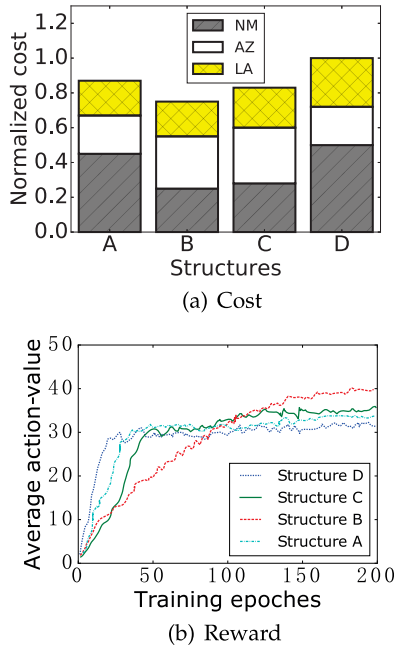

(a) Cost

(b) Reward

Fig. 8. Cost and reward in different discarding ratio of RPS.

comparison, we also show the performance of following algorithms as benchmarks.

- *Round robin (RR)*: It schedules jobs to data centers in a round-robin manner without any information about renewable energy generation [28].
- *MinBrown (MB)*: The MB algorithm is developed for the minimizing brown energy consumption based on workload scheduling strategy [29].
- *Temporal Difference (TD)*: TD is the default algorithm used by traditional RL [15].

5.2 Experimental Results

We analyze the experimental results from the aspect of the affects of network structures, the performance of RPS, the impacts of UBN, the comparison with other algorithms, and the impact of λ and η. Moreover, the convergence of the reinforcement learning is discussed.

5.2.1 The affects of Network Structures

We first study the total cost and convergence speed of our proposed algorithm with different NN structures. In Fig. 7a, we normalize all cost to the result of structure D. The normalized cost of structures A, B, and C is lower than 1, indicating that they outperform D. Structure B has the lowest cost because of the advantages of MLP with larger number of layers. The convergence is shown in Fig. 7b, where we observe that structure D has the fasted converge speed, but leading to a highest cost and lowest average action-value. Meanwhile, structure B converges slower, but it has lowest cost and highest average action-value. Therefore, the structure B can make a good trade-off between cost, convergence speed, and convergence effect, which will be adopted in the following experiments.

5.2.2 The Performance of RPS

We then investigate the performance of RPS by changing the discarding ratio. As shown in Fig. 8a, the lowest cost can be achieved by setting the discarding ratio to 50 percent.
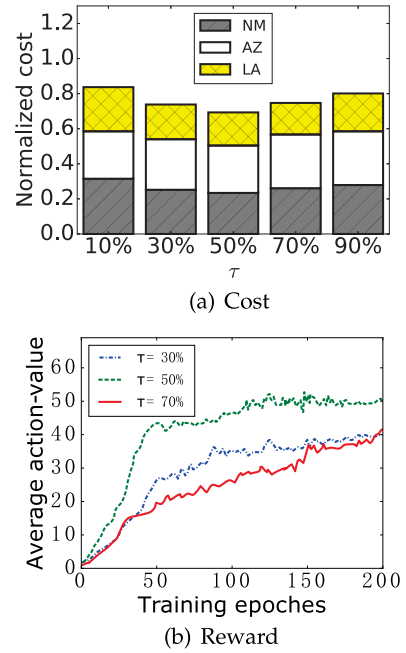
That is because under larger discarding ratio, many data records in the pool will be dropped when it is full, leading to an obvious temporal relation between two mini-batch, i.e., the data in latter mini-batch are always later than that in the previous. On the other hand, retraining based on mini-batch SGD will be frequently triggered under smaller discarding ratio, and a large amount of redundant data will be generated. Furthermore, the results in Fig. 8b demonstrate that setting discarding ratio to 50 percent can guarantee the fastest convergence speed and the highest average action-value. Therefore, when the discarding ratio is set to 50 percent, the RPS converges the best.

5.2.3 The Impacts of UBN

We investigate the acceleration of UBN under different dropout policies. As shown in Fig. 9, the algorithm without UBN slowly converges, but others with dropout has higher average action-value and faster convergence speed, since dropout makes the NN more adaptive by preventing the solution from sinking into a local optimum. UBN with 50 percent dropout has highest average action-value and converges the best.

5.2.4 The Comparison with Other Algorithms

In this set of experiments, we choose structure B with randomly discarding ratio at 50 percent, and a pre-trained
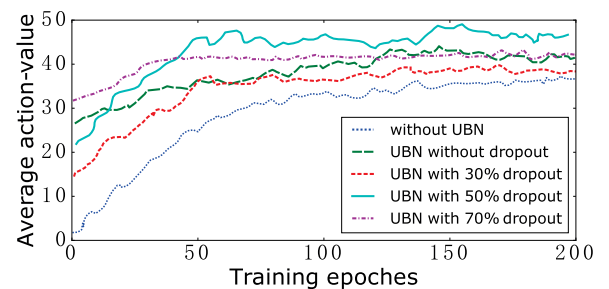

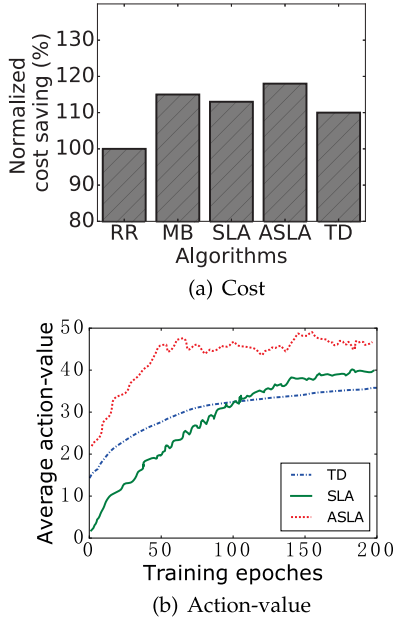Fig. 9. Average action-value of data centers.

(a) Cost

(b) Action-value

Fig. 10. Comparison among ASLA, SLA, and benchmark algorithms.

(a) Different λ

(b) Different η

Fig. 11. Costs of ASLA with different λ and η.

network is connected to the network to be trained using transfer layers with 50 percent dropout. Comparison among our proposed RL-based Algorithm (Strategy Learning Algorithm, SLA), advanced strategy learning algorithm (ASLA) that combines SLA with Algorithm 2, and benchmark algorithms are shown in Fig. 10. As shown in Fig. 10, ASLA has better performance on the normalized average cost saving than most of benchmark algorithms and SLA. In ASLA, the job scheduler has the exploration mechanism controlled by η, which prevent the data centers from an extreme load imbalance. Specially, the average action-value comparison among SLA, ASLA, and TD is shown in Fig. 10b. According to Fig. 10b, SLA and ASLA both perform better than TD that uses tabular action-value function.

5.2.5 The Impact of λ and η of ASLA

The normalized costs of ASLA with different λ and η are shown in Fig. 11. According to Fig. 11a, the costs are basically not affected by the value of λ. We can see that the normalized cost is influenced by η obviously in Fig. 11b. Since the low η controls ASLA to concern more load rather than cost, when the η is set to 0.9, ASLA saves the most cost.

6 RELATED WORK

6.1 Green Data Centers

The energy reduction issue of a single data center are undergoing extensively studied in recent years. For example, Wu et al. [4] have proposed to use dynamic CPU voltage to minimize power consumption of data centers. A similar approach could dynamically adjust the performance of data centers according to the load [5].

The other branch focuses on using task distribution technical across geo-distributed data centers to achieve cost minimization. Rahman et al. [30] presented a survey of geographic load balancing in smart grid. Liu et al. [31] proposed GreenCloud, integrating online monitoring and VM placement optimization. Zhang et al. [32] developed Hierarchical EneRgy Optimization (HERO) scheme for the energy consumption
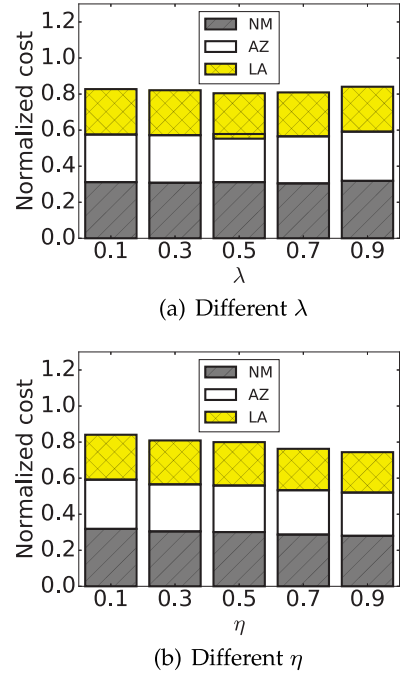
reduction of network equipments. The HERO switches off a part of network switches and connections, which ensures the entire network's connectivity and promotes the links utilization. Adel et al. [28] applied fuzzy inference engine on data centers control. Liu et al. [6] maximized green energy's usage (renewable energy) and minimized the brown energy's usage (fossil fuel energy), respectively.

6.2 Reinforcement Learning

Reinforcement learning attracts more and more attention recent years and it is often used in controlling and routing [33], [34]. Luiz et al. [35] developed a transfer learning based Q-learning approach. Wu et al. [36] proposed to use Q-learning algorithm to optimize the last 2-hops from the source to the destination for long-term efficiency. Mnih et al. [37] successfully combined the deep neural network (DNN) with reinforcement learning and achieved a excellent result in winning game scores. Cui et al. [38] developed an NN-based reinforcement learning algorithm to optimize trajectory tracking for autonomous underwater vehicle (AUV), where a critic NN and an action NN are used. Lange et al. [39] propose to combine deep aoto-encoder NN with batch-mode reinforcement learning in order to deal with the disparity between the visual observation and state space.

However, there are only a few researches of combining reinforcement learning with geographic data centers load balancing. Lin et al. [15] proposed a reinforcement learning-based framework to optimize the power management, but they just uses a simple value function, which may not be able to fit the data centers appropriately.

6.3 Load Balancing

Load balancing is a research hotspot in geo-distributed data centers. Tang et al. [40] proposed a heuristic scheduling approach to balance the workload dynamically. Chen et al. Wang et al. [41] proposed a method of load balancing

towards multimedia big data in data centers. The method is based on hybrid stream and CNN. Chen et al. [42] proposed a network topology aware based approach to perform load balancing, which takes advantage of parallelization and is able to speed up load balancing process. Gupta et al. [43] applied Ant Colony Optimization in load balancing. The algorithm performs load balancing by detecting overloaded and underloaded servers. Shao et al. [44] investigated the optimal load balancing problem and took transmission delay into account.

Different from these previous work, our approach uses reinforcement learning to distribute jobs coming to the data centers, and replaces the value function with NN. Different from these studies, our proposal are self-learn online algorithms. In addition, we design a random pool sampling approach to dismiss the time correlation among rewards and speed up the convergence of the NN. We also propose a new network structure for receive the priori knowledge from other network, which brings significant efficiency improvement. We propose the combination of RL and NN for the first time.

7 CONCLUSION AND FUTURE WORK

In this paper, we have investigated a big data scheduling problem for reducing the cost of geo-distributed data centers. An RL based job scheduling algorithm is proposed with NN, and two techniques are developed to enhance the performance of our proposal. Specifically, we propose RPS to retrain the NN via accumulated training data, and design a novel UBN structure for further enhancing the training speed. Extensive experiments show that our proposal is able to reduce the data centers' cost significantly compared with some benchmark algorithms.

Motivated by this paper, there are many interesting directions that can be studied. With respect to the design of parallel algorithms, one aspect is to use multiple processes to accelerate the learning speed of job scheduler. Another aspect is to investigate the combination of ensemble learning and reinforcement learning for rapid deployment. In the field of decentralized distributed systems, multi-agent reinforcement learning is an aspect can be studied. As for the exploration policy, our approach uses simple greedy algorithm based on probability. However, exploration policy is also important in practice. Further studies will focus on reducing the cost of cloud service.

REFERENCES

[1] K. Wang, Y. Wang, X. Hu, Y. Sun, D.-J. Deng, A. Vinel, and Y. Zhang, "Wireless big data computing in smart grid," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 58–64, Apr. 2017.
[2] P. Li, S. Guo, T. Miyazaki, M. Xie, J. Hu, and W. Zhuang, "Wireless big data computing in smart grid," *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 34–42, Sep. 2016.
[3] Wholesale Electricity and Natural Gas Market Data, (2016). [Online]. Available: http://www.eia.gov/electricity/wholesale/
[4] C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," *Future Generation Comput. Syst.*, vol. 37, no. 7, pp. 141–147, Jul. 2014.
[5] J. Heo, D. Henriksson, X. Liu, and T. Abdelzaher, "Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm case-study," in *Proc. IEEE Int. Real-Time Syst. Symp.*, 2007, pp. 227–238.
[6] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew, "Greening geographical load balancing," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 657–671, Apr. 2015.
[7] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricitymarket environment," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[8] C. Xu, K. Wang, G. Xu, P. Li, S. Guo, and J. Luo, "Making big data open in collaborative edges: A blockchain-based framework with reduced resource requirements," in *Proc. IEEE Int. Conf. Commun.*, May 20–24, 2018, pp. 1–16.
[9] P. Li, S. Guo, T. Miyazaki, X. Liao, H. Jin, A. Y. Zomaya, and K. Wang, "Traffic-aware geo-distributed big data analytics with predictable job completion time," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1785–1796, Jun. 2017.
[10] X. Zhou, K. Wang, W. Jia, and M. Guo, "Reinforcement learning based adaptive resource management of differentiated services in geo-distributed data centers," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service*, 2017, pp. 1–6.
[11] C. Xu, K. Wang, and M. Guo, "Intelligent resource management in blockchain-based cloud datacenters," *IEEE Cloud Comput.*, vol. 4, no. 6, pp. 50–59, Nov./Dec. 2017.
[12] K. Wang, H. Li, Y. Feng, and G. Tian, "Big data analytics for system stability evaluation strategy in the energy internet," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1969–1978, Aug. 2017.
[13] Y. J. Liu, Y. Li, Tang, S. Tong, C. L. P. Chen, and D. J. Li, "Reinforcement learning design-based adaptive tracking control with less learning parameters for nonlinear discrete-time MIMO systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 165–176, Jan. 2015.
[14] X. He, K. Wang, T. Miyazaki, H. Huang, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric IoT," *IEEE Trans. Emerging Topics Comput.*, to be published, doi: 10.1109/TETC.2018.2812788.
[15] X. Lin, Y. Wang, and M. Pedram, "A reinforcement learning-based power management framework for green computing data centers," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2016, pp. 135–138.
[16] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Proc. Int. Joint Conf. Neural Netw.*, 1989, pp. 593–605
[17] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. M. Lau, "Moving big data to the cloud," in *Proc. IEEE INFOCOM*, 2013, pp. 405–409.
[18] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. 34th Annu. Int. Symp. Comput. Archit.*, 2007, pp. 13–23.
[19] Q. Wu, F. Ishikawa, Q. Zhu, and Y. Xia, "Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters," *IEEE Trans. Serv. Comput.*, vol. 12, no. 4, pp. 550–563, Jul.-Aug. 2019.
[20] X. He, K. Wang, H. Huang, and B. Liu, "QoE-driven big data architecture for smart city," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 88–93, Feb. 2018.
[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep reinforcement learning," arXiv:1312.5602, 2013.
[22] K. Wang, C. Xu, Y. Zhang, S. Guo, and A. Y. Zomaya, "Robust big data analytics for electricity price forecasting in the smart grid," *IEEE Trans. Big Data*, vol. 5, no. 1, pp. 34–45, Mar 2019.
[23] T. L. Nwe, N. T. Hieu, and D. K. Limbu, "Bhattacharyya distance based emotional dissimilarity measure for emotion classification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May. 2013, pp. 7512–7516
[24] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
[25] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov 2011, [Online]. Available: http://code.google.com/p/googleclusterdata/wiki/TraceVersion2

[26] National Renewable Energy Laboratory, (2016). [Online]. Available: http://www.nrel.gov/

[27] I. Petrás and M. Gilli, "Complex dynamics in one-dimensional CNNs," *Int. J. Circuit Theory Appl.*, vol. 34, no. 1, pp. 3–20, Jan. 2006.

[28] A. N. Toosi and R. Buyya, "A fuzzy logic-nased controller for cost and energy efficient load balancing in geo-distributed data centers," in *Proc. Int. Conf. Utility Cloud Comput.*, 2015, pp. 186–194.

[29] C. Chen, B. He, and X. Tang, "Green-aware workload scheduling in geographically distributed data centers," in *Proc. Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2012, pp. 82–89

[30] A. Rahman, X. Liu, and F. Kong, "A survey on geographic load balancing based data center power management in the smart grid environment," *IEEE Commun. Surveys Tutorials*, vol. 16, no. 1, pp. 214–233, Feb. 2014.

[31] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen, "GreenCloud: A new architecture for green data center," in *Proc. ACM Int. Conf. Ind. Session Autonomic Comput. Commun. Ind. Session*, 2009, pp. 29–38.

[32] Y. Zhang and N. Ansari, "HERO: Hierarchical energy optimization for data center networks," *IEEE Syst. J.*, vol. 9, no. 2, pp. 406–415, Jun. 2015.

[33] H. A. A. Al-Rawi, K. L. A. Yau, H. Mohamad, N. Ramli, and W. Hashim, "A reinforcement learning-based routing scheme for cognitive radio ad hoc networks," in *Proc. IFIP Wireless Mobile Netw. Conf.*, 2014, pp. 1–8.

[34] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst. Man Cybern.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[35] L. A. C. Jr ., J. P. Matsuura, R. L. de Mantaras, and R. A. C. Bianchi, "Using transfer learning to speed-up reinforcement learning: A cased-based approach," in *Proc. Latin Amer. Robot. Symp. Intell. Robotic Meet.*, 2010, pp. 55–60.

[36] C. Wu, T. Yoshinaga, Y. Ji, T. Murase, and Y. Zhang, "A reinforcement learning-based data storage scheme for vehicular Ad Hoc networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 6336–6348, Jul. 2017.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, and others, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[38] R. Cui, C. Yang, Y. Li, and S. Sharma, "Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning," *IEEE Trans. Syst. Man Cybern.: Syst.*, vol. 47, no. 6, pp. 1019–1029, Jun. 2017.

[39] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw.*, 2010, pp. 1–8.

[40] F. Tang, L. T. Yang, C. Tang, J. Li, and M. Guo, "A dynamical and load-balanced flow scheduling approach for big data centers in clouds," *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 915–928, Oct.-Dec. 2018.

[41] K. Wang, J. Mi, C. Xu, Q. Zhu, L. Shu, and D.-J. Deng, "Real-time load reduction in multimedia big data for mobile Internet," *ACM Trans. Multimedia Comput., Commun. Appl.*, vol. 26, no. 5, Oct. 2016, Art. no. 76.

[42] K. T. Chen, C. Chen, and P. H. Wang, "Network aware load-balancing via parallel VM migration for data centers," in *Proc. Int. Conf. Comput. Commun. Netw.*, Aug. 2014, pp. 1–8.

[43] E. Gupta and V. Deshpande, "A technique based on ant colony optimization for load balancing in cloud data center," in *Proc. Int. Conf. Inf. Technol.*, Dec. 2014, pp. 12–17.

[44] H. Shao, L. Rao, Z. Wang, X. Liu, Z. Wang, and K. Ren, "Optimal load balancing and energy cost management for internet data centers in deregulated electricity markets," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2659–2669, Sep. 2014.

Chenhan Xu is working toward the postgraduate degree in the School of Internet of Things, Nanjing University of Posts and Telecommunications, China. His current research interests include big data, cloud computing, and machine learning.

Kun Wang [M'13-SM'17] received the BEng and PhD degrees in computer science from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004 and 2009, respectively. From 2013 to 2015, he was a postdoc fellow in Electrical Engineering Department, University of California, Los Angeles (UCLA), CA. In 2016, he was a research fellow in the School of Computer Science and Engineering, the University of Aizu, Aizu-Wakamatsu City, Fukushima, Japan. He is currently a research fellow in the Department of Computing, the Hong Kong Polytechnic University, Hong Kong, China, and also a full professor in the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China. He has published more than 100 papers in refereed international conferences and journals. He has received Best Paper Award at IEEE GLOBECOM16. He serves as associate editor of *IEEE Access*, editor of the *Journal of Network and Computer Applications*, the *Journal of Communications and Information Networks*, the *EAI Transactions on Industrial Networks and Intelligent Systems* and guest editors of IEEE Access, Future Generation Computer Systems, Peer-to-Peer Networking and Applications, and Journal of Internet Technology. He was the symposium chair/co-chair of IEEE IECON16, IEEE EEEIC16, IEEE WCSP16, IEEE CNCC17, etc. His current research interests include the area of big data, wireless communications and networking, smart grid, energy Internet, and information security technologies. He is a senior member of the IEEE and Member of ACM.

Peng Li [M'12] received the BS degree from the Huazhong University of Science and Technology, China, in 2007, and the MS and PhD degrees from the University of Aizu, Japan, in 2009 and 2012, respectively. He is currently an associate professor with the University of Aizu, Japan. His research interests include wireless communication and networking, specifically wireless sensor networks, green and energy-efficient mobile networks, and cross-layer optimization for wireless networks. He also has interests on cloud computing, big data processing and smart grid. He is a member of the IEEE.

Rui Xia received the BS degree from Shanghai Jiao Tong University, Shanghai, China, and now is working as an automation developer in Autodesk(China)Software Research and Development Co., Ltd., Shanghai, China. Her research interests include machine learning, big data and web service.
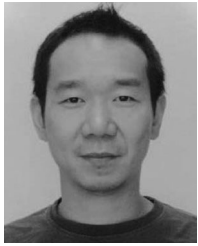
Song Guo [M'02-SM'11] received the PhD degree in computer science from the University of Ottawa and was a professor with the University of Aizu. He is a full professor in the Department of Computing, The Hong Kong Polytechnic University. His research interests include the areas of big data, cloud computing and networking, and distributed systems with more than 400 papers published in major conferences and journals. His work was recognized by the 2016 Annual Best of Computing: Notable Books and Articles in Computing in ACM Computing Reviews. He is the recipient of the 2017 IEEE Systems Journal Annual Best Paper Award and other five Best Paper Awards from IEEE/ACM conferences. He was an associate editor of the *IEEE Transactions on Parallel and Distributed Systems* and an IEEE ComSoc Distinguished Lecturer. He is now on the editorial board of the *IEEE Transactions on Emerging Topics in Computing*, the *IEEE Transactions on Sustainable Computing*, the *IEEE Transactions on Green Communications and Networking*, and the *IEEE Communications*. He also served as General, TPC and Symposium Chair for numerous IEEE conferences. He currently serves as an officer for several IEEE ComSoc Technical Committees and a director in the ComSoc Board of Governors. He is a senior member of the IEEE.

Minyi Guo [F'17] received the PhD degree in computer science from the University of Tsukuba, Tsukuba, Japan. He is currently a zhiyuan chair professor in Shanghai Jiao Tong University, Shanghai, China. His research interests include pervasive computing, parallel and distributed processing, and parallelizing compilers. In 2007, he received the Recruitment Program of Global Experts and Distinguished Young Scholars Award from the National Natural Science Foundation of China. He is on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems* and the *IEEE Transactions on Computers*. He is a fellow of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.