information

Article

Resource Recommender for Cloud-Edge Engineering

Amirmohammad Pasdar [1], Young Choon Lee [1,*], Tahereh Hassanzadeh [2] and Khaled Almi'ani [3,4]

[1] Department of Computing, Macquarie University, Sydney 2109, Australia; amirmohammad.pasdar@hdr.mq.edu.au
[2] School of Engineering and Information Technology, University of New South Wales, Canberra 2610, Australia; t.hassanzadehkoohi@student.unsw.edu.au
[3] Higher Colleges of Technology, Fujairah, United Arab Emirates; kalmiani@hct.ac.ae
[4] Department of Computer Science, Al-Hussein Bin Talal University, Ma'an 71111, Jordan
* Correspondence: young.lee@mq.edu.au

Abstract: The interaction between artificial intelligence (AI), edge, and cloud is a fast-evolving realm in which pushing computation close to the data sources is increasingly adopted. Captured data may be processed locally (i.e., on the edge) or remotely in the clouds where abundant resources are available. While many emerging applications are processed in situ due primarily to their data intensiveness and short-latency requirement, the capacity of edge resources remains limited. As a result, the collaborative use of edge and cloud resources is of great practical importance. Such collaborative use should take into account data privacy, high latency and high bandwidth consumption, and the cost of cloud usage. In this paper, we address the problem of resource allocation for data processing jobs in the edge-cloud environment to optimize cost efficiency. To this end, we develop Cost Efficient Cloud Bursting Scheduler and Recommender (CECBS-R) as an AI-assisted resource allocation framework. In particular, CECBS-R incorporates machine learning techniques such as multi-layer perceptron (MLP) and long short-term memory (LSTM) neural networks. In addition to preserving privacy due to employing edge resources, the edge utility cost plus public cloud billing cycles are adopted for scheduling, and jobs are profiled in the cloud-edge environment to facilitate scheduling through resource recommendations. These recommendations are outputted by the MLP neural network and LSTM for runtime estimation and resource recommendation, respectively. CECBS-R is trained with the scheduling outputs of Facebook and grid workload traces. The experimental results based on unseen workloads show that CECBS-R recommendations achieve a ∼65% cost saving in comparison to an online cost-efficient scheduler (BOS), resource management service (RMS), and an adaptive scheduling algorithm with QoS satisfaction (AsQ).

Keywords: cloud bursting; edge-cloud environment; runtime estimation; recommendation; recurrent neural network (RNN); prediction; long short term memory (LSTM)

check for updates

Citation: Pasdar, A.; Lee, Y.C.; Hassanzadeh, T.; Almi'ani, K. Resource Recommender for Cloud-Edge Engineering. *Information* **2021**, *12*, 224. https://doi.org/10.3390/info12060224

Academic Editors: Flavia C. Delicato, Paulo F. Pires and Kevin I-Kai Wang

Received: 1 May 2021
Accepted: 20 May 2021
Published: 25 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Cost efficiency has been a major driving force for the wide adoption of public clouds, such as Amazon Web Services (AWS) [1], Microsoft Azure (MS) [2], Google Cloud Platform (GCE) [3], and Alibaba Cloud (AC) [4], particularly with the pay-as-you-go pricing. These clouds host servers distributed around the world to provide nearly every Information and Communications Technology (ICT) service (e.g., emails, social media, e-commerce and e-health). While cloud resources are in the core network, pushing computing close to the data sources (i.e., resources residing at the network's edge) is extensively being adopted for the low latency and privacy preserving scheduling of data processing jobs. Scheduling can be further motivated by considering the role of artificial intelligence (AI) in recommending cost-efficient resources in the edge-cloud environment.

There have been many studies, with public clouds and edge-cloud environments (i.e., hybrid clouds), e.g., [5–12], aimed at minimizing resource usage, resulting in cost

reduction with the consideration of some constraints, such as deadline and monetary budget. However, these studies tend to overlook privacy and the utility/energy cost of edge resources. Besides, their performance goals are often achieved with an assumption of prior knowledge of job characteristics (runtimes in particular) [13,14].

While the availability of accurate job information is largely unrealistic, the estimation of such information with reasonable accuracy is possible with the help of recent advances in machine learning. In particular, the recent Microsoft Azure traces [15,16] show that workloads/jobs arriving at a particular cloud often have similar characteristics and resource requirements. Therefore, learning such information through machine learning techniques [17,18] can facilitate edge-cloud scheduling and can be beneficial, not only for runtime estimation, but also for resource recommendations. Hence, we address the problem of cost-efficient job scheduling in an edge-cloud environment while preserving privacy through recommendations. The scheduling not only considers public cloud billing cycles and the edge utility cost, but also relies on runtime estimations for resource recommendation. As an extensive extension to our previous work [19], the CECBS-R framework is assisted by machine learning techniques to facilitate scheduling through runtime estimations and recommendations, as well as improving edge resource utilization while reducing energy consumption.

To this end, we propose Cost Efficient Cloud Bursting Scheduler and Recommender (CECBS-R) framework that is assisted by ANNs, i.e., RNN (recurrent neural network) and LSTM, to deal with scheduling and recommendation while preserving privacy. Figure 1 shows the overall CECBS-R structure. CECBS-R consists of a cloud state controller (CSC) (Algorithm 1) and a cloud scheduler (CS) (Algorithm 2). The Cloud State Controller periodically collects information about jobs in the edge-cloud environment and the edge resource utilization to be processed by ANNs for recommendations. Edge resources (e.g., servers) are constantly monitored, and based on the recommendations, they may become activated or deactivated. The Cloud Scheduler considers feasible recommendations for scheduling jobs through a particle swarm-based algorithm, taking into account the workload's objectives and constraints. Moreover, the MLP neural network employed in the CECBS-R profiles jobs and assists runtime estimation through the collected information in the edge-cloud environment.

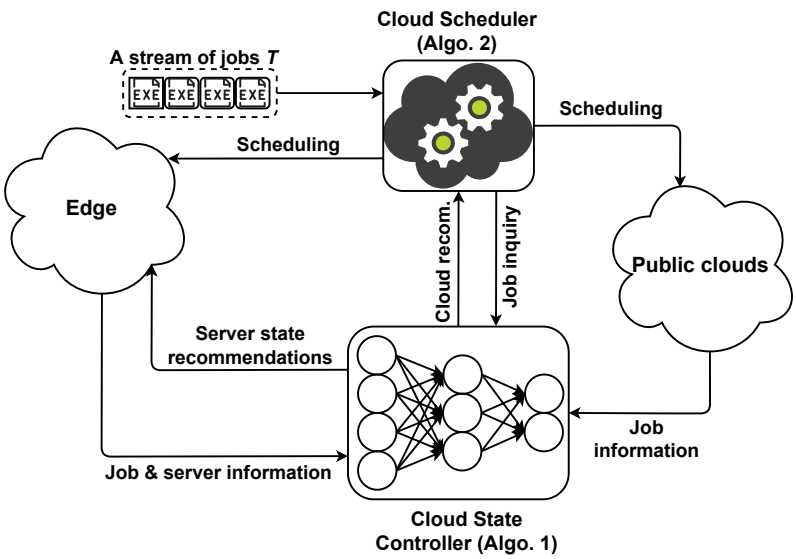


Figure 1. Structure of CECBS-R.

CECBS-R has been evaluated in two phases, based on Facebook workload traces (including synthesized traces), grid workloads [20,21], and the real-world resource requirement traces [15]. The first phase evaluates the cost efficiency of CECBS-R scheduler in comparison to an adaptive scheduling algorithm with QoS satisfaction (AsQ) [22], online

cost-efficient scheduler (BOS) [6], and resource management service (RMS) [5]. In the second phase, CECBS-R is trained by the executed workload information of the previous phase to schedule new workload traces based on the recommendation results. The experimental results show that CECBS-R scheduler on average achieves ∼67% cost-saving in comparison to BOS, RMS, and AsQ. Moreover, recommendations produced by ANNs for a set of new workloads achieved a ∼65% cost saving.

The remainder of the paper is organized as follows. In Section 2, the model and problem statement are defined, followed by Section 3, which explains CECBS-R in detail. Results are discussed in Section 4. We review the related work in Section 5 and conclude this paper in Section 6.

2. Models and Problem Formulation

In this section, the application and system models are described, and the problem is formulated. The symbols used in this paper are described in Table 1.

2.1. The Multi-Cloud

The multi-cloud environment (MC) in this study comprises an edge (R) and P public cloud providers (CP), each of which has a set of resources (R_{CP}).

$$MC = R + \bigcup_{1}^{p} R_{CP} \tag{1}$$

2.1.1. Public Clouds

Resources in public clouds are virtually unlimited, and they are virtualized and called virtual machines (VMs). Each virtual machine (vm_i) is recognized by its computation capacity in terms of vCPUs and memory, storage space, and available bandwidth. Each public cloud (CP_i) maintains a specific billing cycle (BC_i) to determine user resource consumption. Moreover, there is an approximate VM deployment time for each CP_i to have VMs ready.

When a user launches a VM, an active cycle (BC_i^a) becomes available and can be used for free within the active period. Otherwise, a new cycle (BC_i^n) is created that will incur charges. Moreover, each public cloud provider (CP_i) may possess unused resource capacities known as preemptible VMs [3,4], spot instances [1], or low priority VMs [2]. These resources may be offered at a cheaper price and/or based on a specific duration (up to six hours for spot instances [1] or an hour for Alibaba Cloud [4]). Also, each public cloud provider (CP_i) may come with an interruption (i.e., decommissioning) probability for unused resources. This probability describes how often resources may be decommissioned [1].

Moreover, a cloud provider (CP_i) charges users whenever VMs consume storage and bandwidth determined in GigaBytes (GB) during a period. In other words, if a cloud provider (CP_i) charges $\mathbb{B}_i$ per b consumed bandwidth and $\mathbb{S}_i$ per s consumed storage for duration d_s, the expected cost (EC_i) is defined as follows.

$$EC_i = \mathbb{B}_i \times b + \mathbb{S}_i \times s \times d_s \tag{2}$$

To compute the cost of public clouds, we define EC_{CP} in Equation (3) as the summation of all active billing cycles for each public cloud provider ($|BC_i|$) and the cost of consumed storage and bandwidth.

$$EC_{CP} = \sum_{i=1}^{P,|BC_i|} BC_i^a + \sum_{i=1}^{P} EC_i \tag{3}$$

Table 1. Symbol description.

Symbol	Description
MC	An edge-cloud environment
CP	Cloud provider
R/R_{CP}	Set of edge/public cloud resources
BC_i	Billing cycle of cloud provider i
BC_i^a/BC_i^n	Active/New billing cycle of cloud provider i
EC	Estimated cost of a cloud
vm_i	A public cloud virtual machine
m_j	An edge server
UT_i	Utilization of the edge server i
ω_i	Expected wattage usage of edge server i
R_i^t	Required resources of job i
D_i^t	Deadline of job i
RC_i^t	Recommended cloud env. of job i
ART_i^t	Actual runtime of job i
ERT_i^t	Estimated runtime of job i
EC_i^t	Estimated cost of job i
$\mathbb{B}/\mathbb{S}$	Set of cloud provider charges for bandwidth/storage
$\mathbb{A}/\mathbb{I}$	Set of active/idle servers in the edge
d_j	The coarse-grained duration i

2.1.2. The Edge

The edge is a cluster of resources (R) where it has a fixed number of homogeneous servers (m), each of which has storage capacity, a certain number of CPU cores (m_i^{core}), and available memory (m_i^{mem}). Each server (m_i) within the cluster consumes electricity for operating that is measured in watts (ω). Electricity may have volatile charging rates (ω_v) at different times of the day or may follow fixed rates (ω_f) per kilowatt-hour ($k\omega$-h).

Servers may have different utilization rates (UT_i) during workload execution that require a wattage usage (i.e., energy supply) based on the hardware specifications [23]. Hence, the cluster may be controlled by adding ($m_i^{\in}$) or removing servers ($m_i^{\ni}$) for energy consumption management. Equation (4) computes the expected wattage usage per server utilization (UT_i). Each server utilization (UT_i) is bound between a lower utilization (UT_{LB}) and upper utilization (UT_{UB}) which ties the wattage usage to their lower-bound (ω_{UT}^{LB}) and upper-bound (ω_{UT}^{UB}) wattage usage, respectively.

$$\omega_i = \frac{UT_i - UT_{LB}}{UT_{UB} - UT_{LB}} \times (\omega_{UT}^{UB} - \omega_{UT}^{LB}) + \omega_{UT}^{LB} \tag{4}$$

However, Equation (4) presents a linear energy consumption model in comparison to non-linear models in the literature (e.g., [24,25]), Fan et al. [26] showed with empirical results that the linear model could reasonably estimate power consumption.

Hence, if there are k active servers inside the edge, the total edge energy cost (EC_ω) per volatile (or fixed) electricity rates for the period τ are expressed as follows.

$$EC_\omega = \sum_{i=1}^{k,\tau} \omega_i \times \omega_v \tag{5}$$

Considering the edge electricity cost, the total edge-cloud cost is stated as follows.

$$EC_{MC} = EC_\omega + EC_{CP} \tag{6}$$

2.2. Workloads

Workloads are defined as a stream of jobs (T) which become available at time τ at the edge. Each job (t_i) is recognized based on the characteristics it maintains. These characteristics are defined as the job resource requirements (R_i^t), submission time (t_i^τ), deadline (D_i^t), privacy status, recommended cloud environment (RC_i^t), and estimated runtime (ERT_i^t). R_i^t is also expressed as number of CPU cores ($R_i^{t_{core}}$), required memory ($R_i^{t_{mem}}$) and storage ($R_i^{t_s}$).

Each job (t_i) also belongs to a category that correlates D_i^t to the category's characteristic defined as interactive or delay-insensitive [15]. Moreover, jobs in a workload are categorized into three job classes referred to as small, medium, and long-running jobs [15,27].

When a server/VM finishes a job (t_i), the job has an actual runtime (ART_i^t) that is recorded for the runtime estimation of future jobs alongside the job(s) resource requirements. This job (t_i) may be assigned to a feasible cloud recommendation (RC_i^t) and may maintain an estimated cost with respect to either ART_i^t or an estimated runtime (ERT_i^t).

The estimated runtime belongs to a coarse-grained classification with respect to the per-hour billing policy up to 24 h of a day in the form of a one-hot vector. Moreover, there is a controller (γ_i) for a job (t_i) that illustrates the runtime duration within 24 h. This is due to the fact that, if a job may run for a day, it is very likely to run longer [15], thus, although in the literature the runtime may be assumed available, in reality, it is not trivial to estimate the runtime of jobs running for more than a day. Hence, γ_i is defined as follows.

$$(\gamma_i, t_i) = \begin{cases} 0 & ART_i^t \leq 24 \\ 1 & Otherwise. \end{cases} \tag{7}$$

2.2.1. Workload Execution Location

In the absence of the recommended cloud/edge for a job, the execution location heuristically relies on the expected execution cost of a job (EC_i^t). This cost is computed based on the job runtime (e.g., ERT_i^t), but it is not always available.

When a job runtime is not available, the estimated cost per specified coarse-grained duration (d) in the edge-cloud environment is computed. The duration is categorized into four time-period classes which are referred to as 1-h [1–4] and 6-h [1], which represent billing cycles of public clouds, 12-h, and 24-h duration. The 12-h and 24-h are considered due to the volatile electricity rates applied to the utility cost in the edge.

Since the billing cycle cost for the specified duration does not change, it is necessary to estimate the runtime cost in the edge. In other words, how much the cost would be if a job fell into the consecutive rate types that are shown in Figure 2. The consecutive rate would be (1) off-peak → shoulder, (2) shoulder → peak → shoulder, or (3) peak → shoulder → off-peak for 12-h runtime duration. The 24-h duration would fall into any combination of the off-peak, shoulder, and peak rates, and it would take into account all the rates for the cost estimation (see Figure 2).

$$(RC_i^t, EC_i^t) = \min\left(\sum^{\forall i \in d} \min_{\forall j \in BC} EC_{CP}, \sum^{\forall i \in d} EC_\omega\right) \tag{8}$$

When the runtime is not available, Equation (8) returns the minimum cost, which is proven by contradiction in Lemma 1.

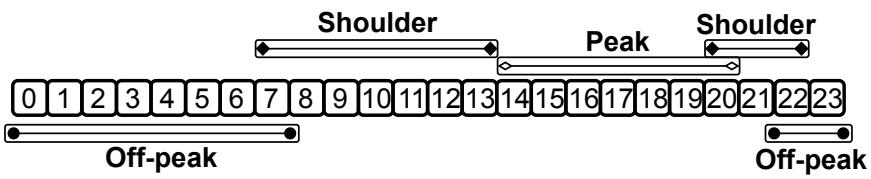


Figure 2. The volatile electricity categories per day stated by Energy Australia [28]. Each square presents a time of day.

Lemma 1. *Given two numbers α and β to represent the edge and public clouds cost computed by Equation (8), respectively, there is a real number c such that $c = \min(\alpha, \beta)$.*

Proof. It is assumed that there is a $c^{'}$; $c^{'} < c$ and it can be written as $c^{'} = \min(\alpha^{'}, \beta^{'})$ in which $\alpha^{'}$ and $\beta^{'}$ are the new minimums.

If $\beta^{'} < \beta$, public cloud billing cycles have volatile rates per specific duration which are against the fact that the rate is fixed, hence, $\beta^{'} = \beta$.

Job t_i becomes available at t_i^{τ} and suppose $\alpha^{'} < \alpha$ which is computed based on four time-period classes; $d_j \in \{d_1, d_6, d_{12}, d_{24}\}$. We replace $\alpha^{'} < \alpha$ considering the job arrival time to compute the expected cost by Equation (5) for each time period.

$$t_i^{\tau} + \{d_1, d_6, d_{12}, d_{24}\} < t_i^{\tau} + \{d_1, d_6, d_{12}, d_{24}\}$$

Since the electricity rate for the off-peak, shoulder, and peak are not volatile, the expected cost for $\alpha^{'}$ and α is identical, which means $\alpha^{'} = \alpha$, thus, $c^{'} = c$. □

When the job runtime (i.e., the estimated runtime (ERT_i^t)) is provided, the expected cost is computed either based on the recommended cloud environment (RC_i^t) or through a cost comparison with respect to ERT_i^t for choosing a cloud/edge environment with minimum cost.

$$(RC_i^t, EC_i^t) = \begin{cases} \min\,(EC_{CP}, EC_{\omega}) & RC_i^t = \emptyset \\ EC_{RC_i^t} & Otherwise. \end{cases} \tag{9}$$

2.3. Problem Statement

CECBS-R for a given stream of tasks T aims to reduce scheduling execution cost while maintaining quality of service (QoS) in the edge-cloud environment.

$$\min_{\forall t \in T} EC_{MC} \tag{10}$$

The scheduling has to take into account the runtime uncertainties which is assisted with ANNs' recommendations. In other words, the problem in our study is two folds. Firstly, how to schedule workloads within the edge-cloud environment when runtime uncertainties exist while preserving privacy in a cost-efficient manner. Secondly, how scheduling can be facilitated with recommendations made by ANNs in the edge-cloud environment.

3. Cost Efficient Cloud Bursting Scheduler and Recommender (CECBS-R)

This section details CECBS-R (Figure 1) components; cloud state controller (Algorithm 1) and cloud scheduler (Algorithm 2), which are assisted with ANNs.

3.1. Cloud State Controller

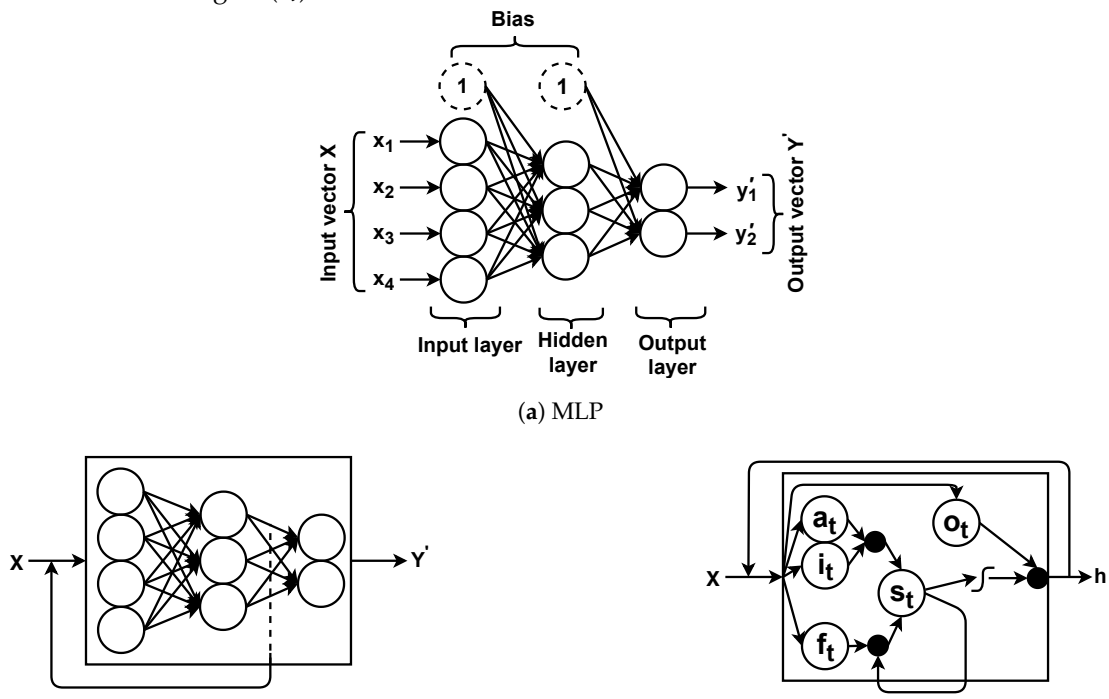
Cloud state controller leverages ANNs for runtime estimation, as well as recommendations to be used for server management in the edge and the job cloud/edge recommendation.

ANNs are computing systems in which examples (i.e., job information or server utilization) are learnt for performing operations, e.g., in terms of cloud environment recommendation or server management. Each neural network consists of connected neurons placed in network layers, and transformations are applied to the value of neurons. Transformations are done by non-linear activation functions to output a value for transmission to the next layer through the network edges. Thus, each value goes through consecutive network layers, starting from an input layer, one layer, or multiple hidden layers and terminating at the output layer. Neurons and edges in a neural network are assigned weights that are adjusted during the network training.

Neural networks used in this study are in different forms referred to as MLP, RNN, or LSTM [29], which are shown in Figure 3.

The MLP is a feed-forward neural network that is trained per each input vector in a supervised manner by backpropagation algorithm with respect to the actual outputs or labels (i.e., $Y = [y_1, \ldots, y_o]$). The MLP repeats the process (e.g., classification) for new inputs, and weights are adjusted accordingly. RNN is a special form of MLP that any learnt (i.e., the hidden layer weights) information is kept and is passed to the next input. In fact, RNNs are a chain of MLPs that leverage shared weights for training. This is to ensure that any prediction/estimation is outputted based on the seen input information.

Although promising, during the learning process, less past/seen information contributes to the training (i.e., weight adjustments) and it is known as a vanishing gradient. This drawback is addressed by LSTMs as the special form of RNNs comprised of gates to control information flow in the learning process. They also have a chain structure consisting of cells, each of which has four gates; forget gate, input gate, input activation gate, and output gate. The information that should be ignored from a cell is dealt with by the forget gate (f_t). To decide what information should be stored in the cell state, there are two gates called input (i_t) and input activation (a_t) gates that contribute to the cell state. Finally, it is required to decide what information should be outputted, which is handled by the output gate (o_t).



(**a**) MLP

(**b**) RNN

(**c**) LSTM

Figure 3. ANN types. (**a**) A three-layer feed-forward network that takes inputs and produces two outputs. (**b**) RNN neural network in which the hidden layer output returns to the network. (**c**) LSTM cell with four gates in which • is Hadamard product and $\int$ is the activation function.

A neural network receives an input vector (i.e., $X = [x_1, \ldots, x_i]$) and outputs an estimated (or interchangeably predicted) output vector (i.e., $Y^{'} = [y_1^{'}, \ldots, y_o^{'}]$) which can be written as $Y^{'} = f(X)$. Neurons in the neural network layers are assigned weights w_{jk}^l that contribute to the neuron output (a_j^l) and are interpreted as the weight from kth neuron in the $(l-1)$th layer to the jth neuron in the lth layer. Moreover, there are extra neurons called bias that hold a value equal to 1 and are connected to the neurons for shifting the

activation function output. Hence, a neuron output (a_j^l) based on sigmoid (σ) (or tangent hyperbolic ($tanh$) or softmax) activation function is defined as follows.

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l) \tag{11}$$

To adjust the neural network weights in ANNs, the actual output vector (Y) is compared with the predicted output (Y') to compute the error (or loss), e.g., mean squared error (MSE). It computes the difference between the N real outputs and the predicted outputs for weight adjustments calculated as follows.

$$E = \frac{\sum_k (y_k^{'} - y_k)^2}{2 \times N} = \frac{\sum_k (a_k^L - y_k)^2}{2 \times N} \tag{12}$$

Without loss of generality, if the neural network weight is W, the weight adjustment with respect to the computed error (E) is defined as follows.

$$W = W - \gamma \nabla_W E \tag{13}$$

In Equation (13), ∇ is the vector differential operator and γ is the learning variable that controls the weight adjustment. The weight adjustment process takes place by the usage of the backpropagation algorithm that performs repetitive procedures in two separate phases; forward pass (ANN_{FW}) and backward pass (ANN_{BW}). In the former, the input data is fed into the network to produce the predicted results with respect to Equation (11). In the latter, the error is calculated in Equation (12) and is propagated back to the neural network to update the network weights.

Considering the ANNs background stated in this section, the RNN is used to control server energy consumption in the edge by autonomously and periodically observing the server utilization. Servers within the edge may be deactivated or activated by a recommended signal. When the signal implies deactivation, a server puts it in deep sleep mode and is added to the idle server list ($\mathbb{I}$). Otherwise, it activates a server from the idle list and is added to the active server list ($\mathbb{A}$). Controlling active servers ($\mathbb{A}$) for reducing energy consumption in the edge requires cluster utilization (i.e., CPU and memory) at and till time τ. This criterion fits in the RNN definition which considers that past seen information for recommendations. Therefore, the RNN is used for the edge server management per specific time intervals to send a signal for activation ($m_i^{\in} \rightarrow \mathbb{A}$) or deactivation ($m_i^{\ni} \rightarrow \mathbb{I}$), in the case of putting servers in a deep sleep mode to reduce energy consumption.

LSTM uses collected information of jobs in the edge-cloud environment for cloud/edge recommendations. The information is job characteristics and resource requirements that autonomously and per specified interval is logged due to the unknown job finish time. LSTM models univariate time series recommendation problems in which single series of observations exist (i.e., the job specifications including the executed edge/cloud environment) with respect to the past observations of jobs to recommend the next value in the sequence. Moreover, compared to the server management, additional information is available for cloud/edge recommendations that necessitates the usage of a sophisticated ANN (i.e., LSTM) for dealing with learning. Although RNN and LSTM rely on the historical context of input, LSTM provides a modulated output controlled by gates facilitating when to learn or forget, and mitigates the downsides of RNNs.

Finally, the job runtime estimation belongs to a coarse-grained classification problem, meaning that the runtime is classified into a time slot class (Section 2.2). Formally, for a job t_j, the classification is interpreted as the time slot γ_j with respect to Equation (7). This classification is controlled by the MLP neural network that uses the collected information about executed jobs across the edge-cloud environment, which are kept as profiles. Moreover, in the absence of a job runtime, the job cloud recommendations will not be reliable. This is due to runtime estimations to tailor the scheduling decision that uncertainty about

the job runtime could blindly lead to an inefficient resource assignment and would affect the quality of scheduling and the cost [13,14].

Algorithm 1 presents an overview of the cloud state controller algorithm. This algorithm consists of procedures with respect to recommendation and runtime estimation, and they interact with the train procedure to obtain the corresponding weights for recommendation purposes.

The train procedure takes input datasets (i.e., the server utilization information and job specifications) and separately trains the corresponding ANN under a specified epoch (lines 1–11). The next procedure controls the edge servers with respect to the given utilization, followed by runtime estimation and recommendation procedures. The algorithm returns the estimated runtime, recommended cloud/edge, and active/idle edge server list.

Algorithm 1: Cloud State Controller (CSC).

Data: Training datasets (DS_X), label datasets (DS_Y), the edge utilization (UT), and jobs T.
Result: RC_T, EST_T, and $\mathbb{A}/\mathbb{I}$

1 **Procedure** `train`(DS_X,DS_Y)
3 Initialize ANN weights (W)
4 **for** *#epoch not exceeded* **do**
5 $\forall \{ds, dy\} \in \{DS_X, DS_Y\}$ **do**
6 $dy^{'} \leftarrow ANN_{FW}(ds, dy, W)$ by Equation (11)
7 Compute $E(dy, dy^{'})$ by Equation (12)
8 $ANN_{BW}(dy, dy^{'}, W)$ by Equation (13)
9 **end**
11 **return** W
1 **Procedure** `control`(UT,W_{UT})
2 $signal \leftarrow ANN(UT, W_{UT})$
3 Increase ($\mathbb{A}$) or decrease ($\mathbb{I}$) active servers
5 **return** $\mathbb{A}$ & $\mathbb{I}$
1 **Procedure** `runtime`(T,W_{ERT})
2 $ERT_T \leftarrow ANN(T, W_{ERT})$
4 **return** ERT_T
1 **Procedure** `recommendation`(T)
2 $RC_T \leftarrow ANN(T, W_{RC})$
4 **return** RC_T
1 *Cloud Scheduler*$(T, RC^T, ERT^T, \mathbb{A}, \mathbb{I})$

3.2. Cloud Scheduler

In this section, we explain how the Cloud Scheduler component (Algorithm 2), which was shown in Figure 1, deals with workload scheduling. The cloud scheduler is a dual-function algorithm that considers runtime estimations for scheduling and may follow recommendations for workload scheduling across the edge-cloud environment.

The cloud scheduler (Algorithm 2) relies on Equation (8) in the absence of cloud recommendations to heuristically examine which environment—the edge or public clouds—would be cost efficient for scheduling. This algorithm illustrates the workload scheduling across the edge-cloud environment which relies on a particle swarm optimization (PSO)-based algorithm for the edge. The algorithm takes into account runtime estimation or cloud/edge recommendations for scheduling decisions. Scheduling in the edge takes place when a job is privacy-sensitive or it is cost efficient to execute regular jobs with respect to active/idle servers states in the edge. Otherwise, these jobs are offloaded to the cost-efficient public cloud.

Algorithm 2: Cloud Scheduler (CS).

Data: T, $\mathbb{A}$, $\mathbb{I}$, RC_T, ERT_T, #generation (G_i), #population (N_p), and PSO parameters
Result: The cost-efficient schedule (CES)

Initialize β, α_T, α_T^b, $V_{\mathbb{P}} \leftarrow 0$, $\{\mathbb{P}, \mathbb{P}_{lc}, \mathbb{P}_{gb}\} \leftarrow \emptyset$
while N_p *not exceeded* **do**
  for $t_i \in T$ **do**
    $R_i^t \leftarrow \{R_i^{t_{mem}} \& R_i^{t_{core}}\}$
    $\alpha_{t_i} \leftarrow \{\forall\ m \in \mathbb{A}\ \exists\ R^m \cap R_i^t \neq \emptyset\}$ & $|\alpha_{t_i}| \leq |T| + 1$
    if $\mathbb{I} \neq \emptyset$ & $|\alpha_{t_i}| \leq 1$ **then**
      $\mathbb{A} \leftarrow \mathbb{I}.poll()$
    end
    if $t_i = regular$ & $\alpha_{t_i} = \emptyset$ & $RC_i^t = R$ **then**
      $RC_i^t \leftarrow \emptyset$
    end
    if $t_i = regular$ & $RC_i^t \neq R$ **then**
      Select RC_i^t w.r.t. Equation (8)
      Update $\mathbb{A}$, α_{t_i}, $\alpha_{t_i}^b$, & β
    end
  end
  $\mathbb{P} \leftarrow RS(T, \alpha_T)$ & $f_{\mathbb{P}}$
  if $\alpha_T = \emptyset$ **then**
    $\alpha_T \leftarrow \alpha_T^b$
  end
  $Shuffle(\alpha_T.poll()$ & update β
end
while G_i *not exceeded* **do**
  Update $\mathbb{P}_{lc}$ and $\mathbb{P}_{gb}$
  for $\rho_i \in \mathbb{P}$ **do**
    Update V_{ρ_i}, L_{ρ_i}, and f_{ρ_i}
    Keep track of CES_ρ
  end
end
Update β, & $\mathbb{A}$ w.r.t. CES
Return CES

Meta-heuristic algorithms such as PSO in this study are considered to deal with many complex scheduling problems, such as task scheduling. However they do not guarantee to find optimal solutions—they can be "effective". The particle swarm optimization algorithm through repetition improves the quality of a candidate solution known as particles (ρ). It aims to move these particles around in the search space based on the position (L_ρ) and the velocity of particles (V_ρ). Each particle's movement is affected by the local ($\mathbb{P}_{lb}$) and the global ($\mathbb{P}_{gb}$) best-known positions toward the best-known positions in the search space (i.e., the population ($\mathbb{P}$)). It eventually leads to moving the population (i.e., swarm) toward the best solutions.

A particle structure in the algorithm that is shown in Figure 4 has a dynamic length equal to the number of available jobs at time τ, and each job is assigned to either an edge server (m_i) or a public cloud virtual machine (vm_j). This selection relies on a candidate list (α_{t_i}) per each job resource requirement (R_i^t).

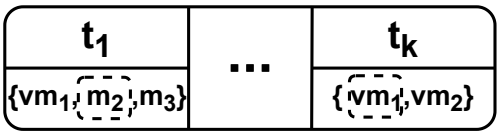


Figure 4. The particle structure. The dashed square is the selected resource.

The initial population ($\mathbb{P}$) is begun with an eligible permutation of job resource selection. There is a list (β) that maintains resource availability across the edge-cloud environment. Each job in the workload (T) is checked for preparing a list of servers and/or virtual machines that can host a job. The list relies on the edge utilization level in which edge servers are descendingly sorted based on the utilization level (both CPU and memory) at time τ. The list of each job is capped at $|T| + 1$ servers to avoid computation overhead caused by PSO. Moreover, a job candidate list is updated by adding a temporarily idle server from the $\mathbb{I}$ list of the edge. This added server would be confirmed if current active servers could not satisfy a job's resource requirement. Moreover, if a job is privacy-sensitive, it has to choose the edge servers. Otherwise, both edge servers and public cloud virtual machines are taken into account.

When CECBS-R works as the scheduler, Algorithm 2 considers estimated runtimes to assist the selection of cost-efficient resources. Otherwise, it will seek feasible recommendations to schedule jobs onto resources in the edge-cloud environment. In other words, if a regular job is recommended to be scheduled onto the edge with no available servers, the recommended cloud is updated based on the most cost-efficient resources with respect to Equation (8) that considers billing cycles, and the estimated runtime (ERT_i^t) across the edge-cloud environment.

Per each resource candidate list of a job (α_{t_i}), the population is updated by mapping a selected resource to a job for forming a solution. Each solution is evaluated against a fitness function for cost-efficiency and the quality of resource assignment. When a job (t_i) is assigned to a candidate resource (m or vm), its resource candidate list (α_{t_i}) and the resource availability list (β) are updated. If during the process, α_{t_i} becomes empty, the backup candidate list ($\alpha_{t_i}^b$) will update the list.

The quality of each particle (ρ) is assessed by the fitness function (f_ρ) that consists of controlling and quality parameters, each of which evaluates a particular aspect of the particle. Controlling parameters are defined as the cloud priority (ρ_ϱ) and the job resource allocation (ρ_ς).

A particle may have jobs that are assigned to public cloud virtual machines, however, there should be a mechanism to prioritize edge resource selection. In other words, jobs should be avoided to be processed on the public clouds while the edge has sufficient resources available. Hence, ρ_ϱ checks jobs in the particle and penalizes the fitness with $\psi \in [0, 1]$, in which the edge has lower penalized value compared to public clouds, i.e., $\psi_R < \psi_{R_{CP}}$.

$$\rho_\varrho = \sum_{i=0}^{|\rho|} \psi_i \tag{14}$$

Moreover, jobs in a particle should be checked, whether or not the chosen server (m) or the virtual machine (vm) can satisfy jobs resource requirements (R_i^t). It is expressed as ρ_ς that returns $true$ or $false$ if the resource requirement is satisfied or not, respectively.

The quality parameters should control the edge utilization (ρ_κ) and the estimated cost (ρ_ζ). If jobs are assigned to servers in the edge, it should have to increase the overall utilization. Therefore, a particle (ρ) is assessed against how chosen servers (m_i) can contribute toward better utilization.

$$\rho_\kappa = \sum_{i=0}^{|\rho|} UT_i \tag{15}$$

Finally, the estimated cost of a particle (ρ) for cost-efficiency is computed in Equation (16). The cost is divided into two parts; the edge and public clouds. The former cost is directly affected by the impact of utilization level on the energy consumption as the higher utilization leads towards the higher energy consumption, and consequently, the higher fitness ratio. In contrast, the latter has a reverse impact on the fitness function, since it aims to reduce the reliance on public cloud resources.

$$(\rho_{\zeta}^{\omega}, \rho_{\zeta}^{EC}) = \begin{cases} \sum_{i=0}^{|\rho|} EC_{\omega} & R_i^t \in R. \\ \sum_{i=0}^{|\rho|} EC_{CP} & Otherwise. \end{cases} \tag{16}$$

If a server of the edge is assigned to a job (t_i), ρ_{ζ}^{ω} will consider the electricity cost based on the utilization that the server will have. Otherwise, it will compute the usage cost of virtual machines in public clouds based on the billing cycle.

The fitness function based on the controlling and quality parameters is defined as follows.

$$f_{\rho} = \frac{\rho_{\varsigma} \times \rho_{\kappa} \times \rho_{\zeta}^{\omega}}{\rho_{\varrho} \times \rho_{\zeta}^{EC}} \tag{17}$$

In the fitness function, if the denominator is increased, the fitness function will lead toward particles that are not cost efficient. This is due to leaving the edge resources under-utilized and relying more on public clouds. If the numerator is increased and becomes aligned with a small denominator, the fitness function will lead to expressing solutions that are cost efficient and will improve the edge resource utilization.

The PSO algorithm through each iteration updates the best local ($\mathbb{P}_{lc}$) and global ($\mathbb{P}_{gb}$) known positions. These positions are the chosen resource indexes in the corresponding candidate list (α_{t_i}) shown in Figure 5. In order to generate new solutions based on the current population, particles should have to move in the search space with respect to the swarm movement terminology. The movement is controlled by a particle location (L_{ρ_i}) and its speed (V_{ρ_i}). Each particle updates its location (L_{ρ_i}) and its velocity (V_{ρ_i}) with respect to the $\mathbb{P}_{lc}$ and $\mathbb{P}_{gb}$, stated in Equations (18) and (19), respectively. The velocity of a particle (V_{ρ_i}) is also controlled by the PSO learning parameters referred to as c_1, c_2, and $\{\xi, \theta\} \in [0, 1]$ [30].

$$\begin{aligned} V_{\rho_i} = \theta \times V_{\rho_i} + c_1 \times \xi \times (\mathbb{P}_{lc} - L_{\rho_i}) \\ + c_2 \times \xi \times (\mathbb{P}_{gb} - L_{\rho_i}) \end{aligned} \tag{18}$$

$$L_{\rho_i} = L_{\rho_i} + V_{\rho_i} \tag{19}$$

(**a**) (**b**)

Figure 5. A t_i candidate list (α_{t_i}) with 5 edge servers $\{m_a, m_b, m_v, m_f, m_e\}$ and one public cloud VM (vm_j). (**a**) Already chosen index before updating the location. (**b**) After updating the location considering 2 for $V_{\rho_j}^{t_i}$.

Velocities in the cloud scheduler (Algorithm 2) are the candidate index positions in α_{t_i} for job t_i. Hence, locations are bounded in the length of $|\alpha_{t_i}|$ that is updated per new index position provided by the new velocity.

$$L_{\rho_j}^{t_i} = \begin{cases} L_{\rho_j}^{t_i} + V_{\rho_j}^{t_i} & V_{\rho_j} \leq |\alpha_{t_i}| \\ L_{\rho_j}^{t_i} + V_{\rho_j}^{t_i} \ \% \ |\alpha_{t_i}| & Otherwise. \end{cases} \tag{20}$$

Equation (20) shows that when the updated velocity ($V_{\rho_j}^{t_i}$) in ρ_j exceeds the length of α_{t_i}, it will return the reminder as the new index for choosing a new candidate. Figure 5 illustrates that if the current index for job t_i in α_{t_i} is 2, the updated velocity $V_{\rho_j}^{t_i} \in \{|\alpha_{t_i}| \times n\}$ ($n \in \mathbb{N}$) will change the location.

3.3. CECBS-R Time Complexity

CECBS-R employs a PSO algorithm for scheduling jobs in the edge that consults with the cloud state controller algorithm to seek recommendations/classification. The PSO fitness function affects the algorithm running time, which relies on the number of jobs available at time τ that also impacts the population size. In other words, if the PSO has N_p populations to become evaluated in G_i iteration, the PSO will converge in $O(N_p \cdot G_i)$ steps.

Moreover, the worst-case upper bound of running time for mapping recommended jobs (l_{off}) to cost-efficient public clouds depends on the cost comparison between the clouds. If there are $|CP|$ public clouds, the running time is stated as $O(|l_{off}| \cdot |CP|)$. Hence, CECBS-R requires the following running time for T jobs at time τ:

$$\max_{T}(O(N_p \cdot G_i), O(|l_{off}| \cdot |CP|))$$

4. Evaluation

In this section, we evaluate the performance of the CECBS-R framework in terms of server usage, QoS violations (i.e., missing deadlines), and cost in comparison to BOS [6], RMS [5], and AsQ [22]. For the baseline algorithms, actual runtimes are available in advance. BOS is a policy-driven hybrid scheduler to maximize utilization while reducing the usage cost of public clouds. It uses two queue management policies, such as first-come-first-served (FCFS) and earliest deadline first (EDF), which are accompanied by two offloading policies referred to as unfeasible and the cheapest jobs to the public clouds. The EDF queue policy, in conjunction with the unfeasible offloading policy, is used for comparison. RMS also maintains the job queues which are managed by heterogeneous earliest finish time (HEFT) algorithm for cost-efficient resource allocation to the jobs leading to performance improvement. AsQ is an optimization algorithm that aims at scheduling deadline-constrained jobs to improve utilization while reducing public cloud rental costs.

The comparison is reported in two separate sections based on the edge server usage, cost, and QoS violations. The first section compares the CECBS-R scheduler against baseline algorithms, and the results are used to build up datasets for training the recommender. The second section shows the result of the CECBS-R recommender compared to the mentioned algorithms.

4.1. Simulation Setup

The edge-cloud in our experiment consists of an edge and four recognized public cloud providers; Amazon Web Services, Microsoft Azure, Google Cloud Engine, and Alibaba Cloud. Public cloud VMs are chosen on-demand from the US East region and their types are selected from general-purpose and compute-optimized VMs. The VMs are subject to charges based on the billing policy (i.e., per hour vs. per second), and this charge considers bandwidth and storage usage. The deployment time of each public cloud provider on average is 100 s for AWS [31], 90 s for Microsoft Azure [32], 90 s for Alibaba cloud, and 54 s for Google Cloud Engine [33].

It is assumed that the edge has a cluster of 200 servers similar to a large private IBM cloud infrastructure [34]. Each server has 64GB memory and a dual processor (18 cores) running at 3300 million instructions per second (MIPS). Moreover, each server under different utilization levels requires an estimated wattage usage that is shown in Table 2. Off-peak, shoulder, and peak electricity rates are \$12.08196, \$51.04627, and \$24.44365 kilowatts per hour [35], respectively.

Table 2. Utilization wattage requirement (*watt*) [23].

0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
188	223	258	293	328	363	398	433	468	503	538

We use Facebook Hadoop traces [20] and public grid workload traces [21] as inputs for our simulation, and jobs are synthesized to be short, medium, and long-running jobs with a ratio of 75:20:5 [15]. Jobs are assigned a length in MIPS and follow a Gaussian distribution with respect to (mean, variance) values that are (1, 10), (10, 100), and (100, 1000) for short, medium, long-running jobs, respectively. Resource requirements for jobs are shown in Table 3 and jobs are categorized into interactive and delay-insensitive groups with 27:73 ratio [15] under relaxed user-defined deadline ratios chosen from [0.1, 0.5] and [0.5, 1.0], respectively. Moreover, jobs follow non-preemptive scheduling, and results are reported for non-privacy jobs and 10% of interactive jobs to be privacy-sensitive.

Table 3. Resource requirements [17].

(a) #Cores.									
Job percentage	<55%	<20%	<10%	<3%	<4%	<2%	<3%	<2%	<1%
CPU cores	1	2	4	6	8	12	20	24	32
(b) Memory (GB).									
Job percentage	<5%	<38%	<4%	<20%	<3%	<10%	<2%	<10%	<8%
Memory	0.75	1.75	2	3.50	4	7	8	14	16/28/32–64

Particle swarm optimization parameters are considered the default values; 2 for both c_1, c_2, and θ is 0.8 and 5% of edge servers are assumed to be initially active. The initial population size is dynamic and is assumed to be a coefficient of incoming jobs and their chosen resources at time τ capped at 200. The generation size (G_i) is considered 200.

4.1.1. Server Management Datasets & Parameters

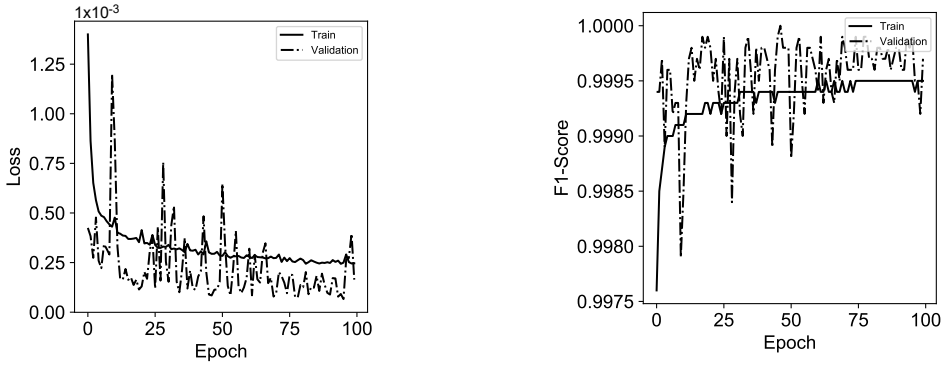
CECBS-R uses virtual machine utilization traces to train the RNN-based server management implemented in Keras; the Python deep learning API [36], and traces are chosen distributed data center performance metrics from Bitbrains composed of 1750 VMs [37]. CECBS-R uses the time interval 60 s for the edge server states and 300 s for checking the job state in the edge-cloud environment [15]. Although VMware introduces 30 s (low), 60 s (medium), and 120 s (high) time intervals, it states that the high and low sensitivity intervals may cause a false or long detection [38]. The RNN has three layers in which the hidden layer has 64 neurons. The input of the RNN is the CPU and memory utilization, as well as overall server utilization that each input belongs to a utilization category; low (under 20%), medium (between 20% and 80%), and high (higher than 80%) [15].

The RNN leverages hyperbolic tangent activation function (*tanh*) for the hidden layer and softmax for the output layer. Furthermore, the training dataset is divided into two sets—training and validation—with a ratio of 80:20 under 100 epochs.

To validate the RNN model for server management, F1-score is reported as a measure of accuracy. The value considers precision and recall for computing the score. The former reports the ratio of correct positive divided by all positive results returned by the classifier. The latter is the ratio of correct positive results divided by the number of all relevant samples. In essence, the higher F1-score emphasizes how perfect the model is. In addition, the loss is reported that shows how well the ANN performs on the training and validation datasets.

Figure 6 shows the overview of training loss and F1-score over 100 epochs (i.e., based on subtle changes in the error ratio) and illustrates the significant accuracy of the model.

The F1-score for the server management proves the performance of the designed classifiers; increasing or decreasing the edge servers with respect to the servers' load.



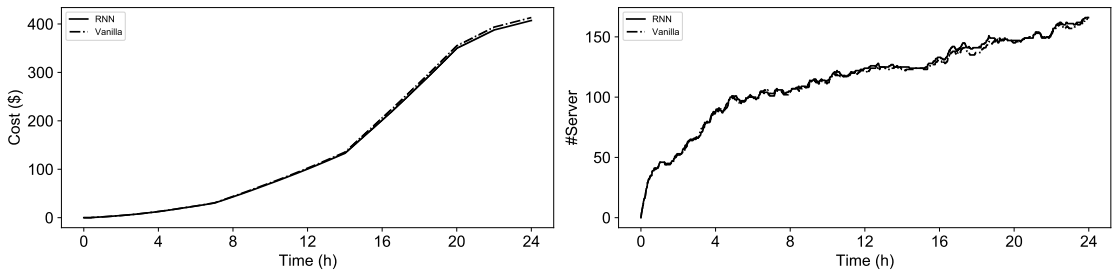
(**a**) Training error (**b**) F1-score

Figure 6. The loss and F1-score over 100 epochs for the RNN server management.

4.1.2. RNN versus Vanilla Server Management

To show how effective the ANN-assisted server management is, a comparison between the RNN model and a vanilla strategy for controlling the servers in the edge is reported. The vanilla strategy takes into account the CPU utilization level at the specified interval and sends a signal with respect to the server utilization level.

We ran CECBS-R based on a crafted trace consisting of ∼13.3 k jobs under the privacy of 10% in which jobs inherit Facebook dataset characteristics, but with a different job arrival time for 24 h. The comparison based on the edge cost and server usage is depicted in Figure 7.



(**a**) Edge cost (**b**) Server usage

Figure 7. RNN versus vanilla server management.

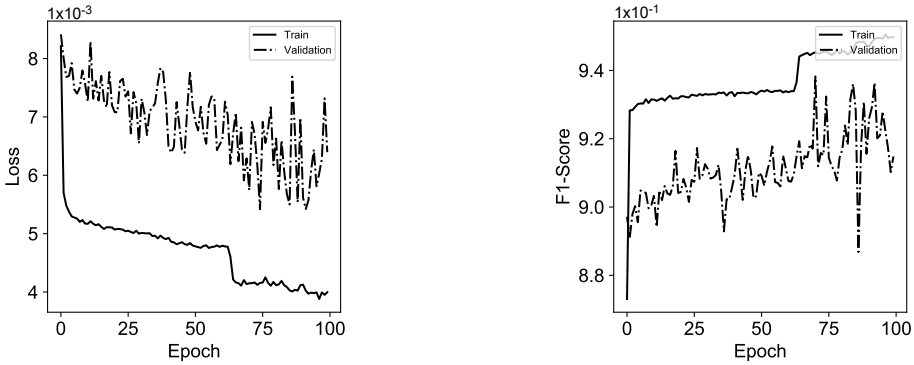
Figure 7a illustrates that the vanilla server management gradually became expensive during the execution time (i.e., ∼3% more expensive), with average costs of $407 and $413 for RNN and vanilla, respectively. Moreover, although they almost achieved the same server usage (on average 135), Figure 7b shows that RNN followed more steady usage patterns than vanilla, resulting in cheaper execution cost for the edge. Recall that the RNN-assisted server management takes into account both CPU, RAM, and overall cluster utilization for the purpose of sending a controlling signal. In contrast, vanilla server management only considers CPU utilization, which could blindly turn on or off servers.

4.1.3. Runtime Estimation Datasets and Parameters

The runtime estimation model employs a four-layer MLP neural network with *tanh* activation function. The input layer receives job characteristics, such as requested memory and CPU core, the length, actual runtime, and the CPU speed of the server or VM where it is executed. The MLP has 64 neurons for hidden layers, which are followed by the output

layer. It is a one-hot vector associated with the duration controller (γ_i) that represents a coarse-grained classification.

The MLP neural network is trained under 100 epochs based on grid workload datasets which consist of more than 1 million jobs logged for two months [39]. CECBS-R relies on the ∼5-day of unseen datasets to collect job execution information across the edge-cloud to be fed into the network (recall Section 2.2.1). Figure 8 shows the overview of training loss and F1-score which is followed by the validation's values. The loss provides a small model error that shows the data is perfectly fitted into the designed model for the runtime estimation. Moreover, the F1-score is high, meaning low false positives and false negatives identified in this model training.



(**a**) Training error (**b**) F1-score

Figure 8. The loss and F1-score over 100 epochs for the runtime estimation model.

4.2. The Scheduler Results

This section presents results for workload scheduling by CECBS-R compared to the baseline algorithms. CECBS-R heuristically schedules jobs across the edge-cloud environment, taking into account runtime uncertainties managed by the MLP neural network. In contrast, the baseline algorithms considering virtually accurate runtimes are available in advance.

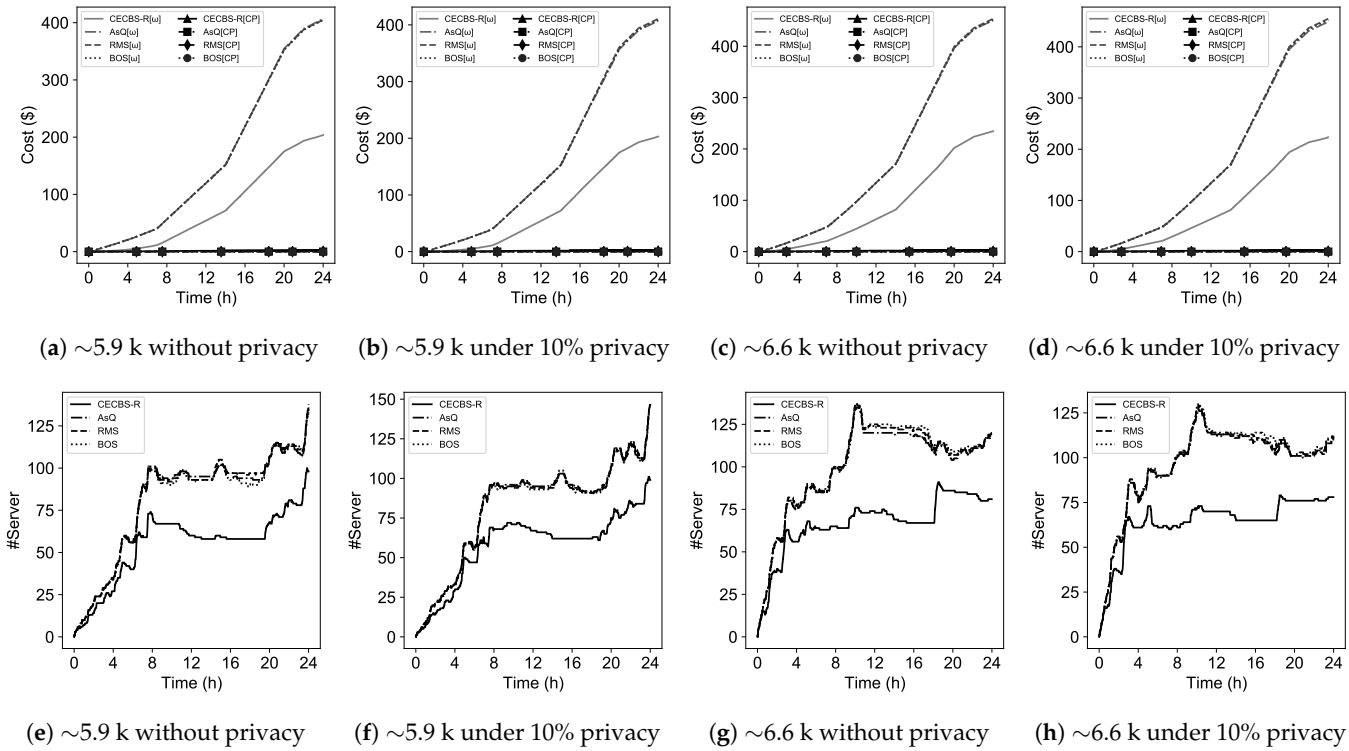
Table 4 presents different workloads' characteristics that are used for this section. The traces contain job information and resource requirements unless otherwise it is stated. The synthesized workloads are crafted based on the measured arrival time between consecutive jobs in the real Facebook traces; ∼5.9k and ∼6.6k. The average arrival time between jobs is reported as 14, hence, to craft new workloads out of the real workloads, the arrival time is chosen from the following set—$\{7, 10, 14\}$—which creates ∼11.6 k, ∼15.8 k, and ∼21.7 k synthesized traces. These synthesized traces are assigned new resource requirements with respect to Table 3. Moreover, CECBS-R reports the results of another public trace workload composed of nearly 4 consecutive days of job submission [21].

Table 4. Workload traces.

Trace	Type	# Jobs in Trace	Duration
Facebook	Real	∼5.9 k	24 h
Facebook	Real	∼6.6 k	24 h
Facebook	Synthesized	∼11.6 k	24 h
Facebook	Synthesized	∼15.8 k	24 h
Facebook	Synthesized	∼21.7 k	24 h
Grid	Real	∼61k	∼4 days

We start by presenting the accumulated edge and public cloud cost for traces ∼5.9 k and ∼6.6 k during the execution time of a day followed by the edge server usage. Figure 9 shows the overall accumulated cost when jobs have no privacy and have a privacy ratio

of 10% along with edge server usage. This figure illustrates that the accumulated cost is impacted by increasing the number of jobs, the privacy ratio, and the corresponding type of jobs. Comparing Figure 9a,b shows that CECBS-R achieved nearly the same edge cost, which is less than the cost of the other algorithms. This is supported by Figure 9e,f. In these figures, CECBS-R utilized the edge servers aligned with the workload assisted by the machine learning server management, taking advantage of public clouds, in particular, AWS. While AsQ and BOS almost have the same cost and less cost than RMS, CECBS-R maintained cheaper workload execution. In Figure 9c, CECBS-R also maintained a lower cost than the baseline algorithms which can be supported by Table 5 as CECBS-R leveraged public clouds (i.e., AWS) resources for workload execution. Furthermore, Figure 9g,h motivate the fact that controlling the edge server usage with respect to the incoming workload can significantly improve the overall edge utilization. For a higher privacy ratio, CECBS-R sent off jobs to the public clouds compared to the other algorithms led to achieving the less cost shown in Figure 9d and Table 5. This table shows the total execution cost across the edge-cloud environment and the contribution of each/cloud environment to this cost. It is understood that the edge cost has the highest contribution to the cost, and the value is the minimum for CECBS-R.



(**a**) ∼5.9 k without privacy (**b**) ∼5.9 k under 10% privacy (**c**) ∼6.6 k without privacy (**d**) ∼6.6 k under 10% privacy

(**e**) ∼5.9 k without privacy (**f**) ∼5.9 k under 10% privacy (**g**) ∼6.6 k without privacy (**h**) ∼6.6 k under 10% privacy

Figure 9. The accumulated edge (ω) and public cloud (CP) cost followed by the edge server usage for ∼5.9 k and ∼6.6 k jobs.

Increasing the number of jobs in traces (Figure 10) changed the accumulated cost across the edge-cloud environment, as well as the edge server usage. The usage of public clouds also became more obvious, in particular, for Figure 10c,d. This shows that CECBS-R achieved the least total edge-cloud cost in comparison to the other algorithms through maintaining the execution workload within the edge. It is expected that it will result in higher server usage, which is clear in Figure 10e–h. Increasing the privacy ratio made the internal resources on the edge busier, therefore, it would be more cost efficient to schedule jobs on the resources across the edge-cloud environment. While CECBS-R barely had information about the accurate runtimes in comparison to the baseline algorithm, it could achieve lower edge cost (Figure 10a,b), which is also aligned with the server usage in Figure 10e,f. Furthermore, in Table 5(a), CECBS-R, on average, utilized more edge resources, leading to maintaining lower public cloud cost, which emphasizes better

resource allocation for the privacy ratios. Increasing the server utilization while the idle servers become deactivated (i.e., put in deep sleep mode) reduced the edge cost significantly for CECBS-R.

Table 5. Edge-cloud job proportion and its cost under volatile (ω^v) electricity rates.

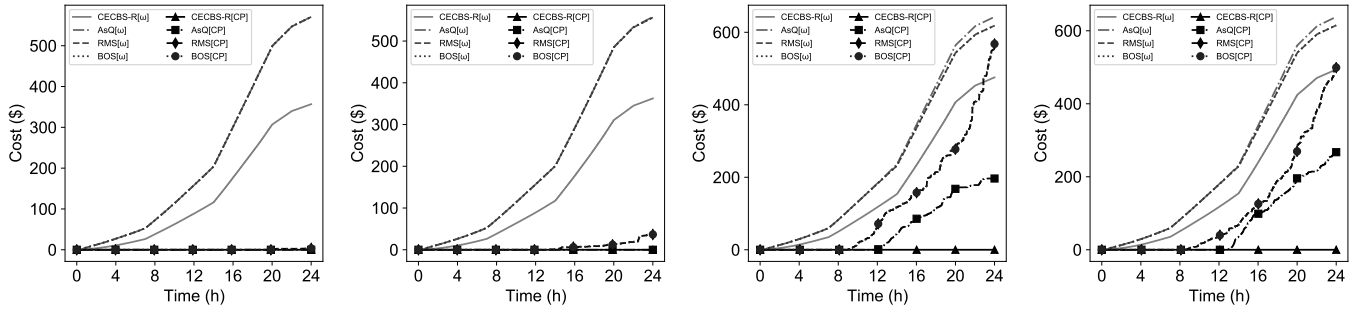
(a) Edge									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼5.9 k	–	5760	5895	5895	5895	203.82	406.98	405.44	404.71
∼5.9 k	10%	5767	5895	5895	5895	203.11	407.20	410.89	408.80
∼6.6 k	–	6492	6639	6639	6639	234.71	450.70	453.69	451.30
∼6.6 k	10%	6488	6639	6639	6639	223.37	448.92	454.93	452.17
∼11.6 k	–	11,578	11,578	11,568	11,568	356.71	570.73	570.63	570.63
∼11.6 k	10%	11,578	11,578	11,503	11,503	362.61	557.97	556.49	556.49
∼15.8 k	–	15,774	12,862	15,109	15,109	475.61	642.62	618.36	618.36
∼15.8 k	10%	15,774	12,683	14,901	14,901	493.94	638.16	614.92	614.92
∼21.7 k	–	21,234	15,315	19,286	19,286	604.29	661.35	642.78	642.78
∼21.7 k	10%	21,265	15,336	18,473	18,473	609.74	660.35	637.67	637.67
∼61 k	–	53,128	34,952	45,975	45,522	2182.22	2437.22	2378.28	2382.69
∼61 k	10%	52,485	37,900	33,224	33,209	2189.01	2439.09	2389.77	2381.63
(b) AWS									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼5.9 k	–	135	0	0	0	2.79	0	0	0
∼5.9 k	10%	126	0	0	0	2.96	0	0	0
∼6.6 k	–	147	0	0	0	3.11	0	0	0
∼6.6 k	10%	151	0	0	0	3.37	0	0	0
∼11.6 k	–	0	0	0	0	0	0	0	0
∼11.6 k	10%	0	0	0	0	0	0	0	0
∼15.8 k	–	0	469	6	6	0	24.2	1.67	1.67
∼15.8 k	10%	0	428	12	12	0	4.93	7.26	7.26
∼21.7 k	–	21	1389	44	44	8.91	47.99	19.01	19.01
∼21.7 k	10%	6	1256	85	85	4.29	16.23	25.96	25.96
∼61 k	–	4006	3818	699	733	224.17	230.08	194.79	168.03
∼61 k	10%	4006	3752	2332	2159	218.26	265.61	187.25	185.36
(c) GCE									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼5.9 k	–	0	0	0	0	0	0	0	0
∼5.9 k	10%	0	0	0	0	0	0	0	0
∼6.6 k	–	0	0	0	0	0	0	0	0
∼6.6 k	10%	0	0	0	0	0	0	0	0
∼11.6 k	–	0	0	0	0	0	0	0	0
∼11.6 k	10%	0	0	2	2	0	0	1.28	1.28
∼15.8 k	–	0	21	10	10	0	12.52	10.47	10.47
∼15.8 k	10%	0	35	22	22	0	30.37	40.43	40.43
∼21.7 k	–	0	39	36	36	0	27.28	74.4	74.4
∼21.7 k	10%	0	68	47	47	0	55.19	96.58	96.58
∼61 k	–	127	200	154	155	64.43	125.73	335.77	328.67
∼61 k	10%	193	213	218	225	105.45	317.79	356.81	384.11

Table 5. *Cont.*

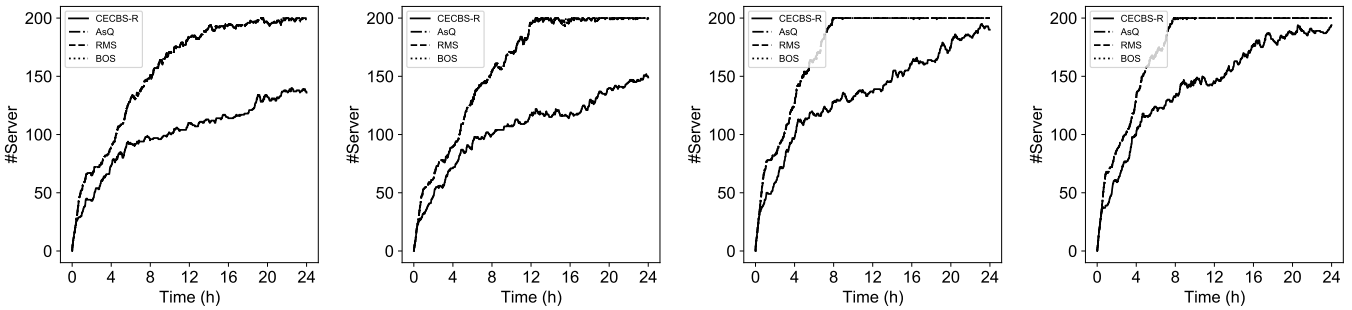
(d) MS									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼5.9 k	–	0	0	0	0	0	0	0	0
∼5.9 k	10%	0	0	0	0	0	0	0	0
∼6.6 k	–	0	0	0	0	0	0	0	0
∼6.6 k	10%	0	0	0	0	0	0	0	0
∼11.6 k	–	0	0	8	8	0	0	1.92	1.92
∼11.6 k	10%	0	0	58	58	0	0	24.53	24.53
∼15.8 k	–	0	2334	464	464	0	70.46	169.77	169.77
∼15.8 k	10%	0	2438	630	630	0	54.35	162.13	162.13
∼21.7 k	–	316	4695	1825	1825	350.28	134.87	290.04	290.04
∼21.7 k	10%	314	4594	2529	2529	458.64	148.37	384.22	384.22
∼61 k	–	1962	20,942	12,220	12,665	3885.6	2170.28	1987.23	1970.62
∼61 k	10%	2168	17,235	23,006	23,208	4156.68	1283.98	1645.17	1728.8
(e) AC									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼5.9 k	–	0	0	0	0	0	0	0	0
∼5.9 k	10%	2	0	0	0	0.14	0	0	0
∼6.6 k	–	0	0	0	0	0	0	0	0
∼6.6 k	10%	0	0	0	0	0	0	0	0
∼11.6 k	–	0	0	2	2	0	0	1.43	1.43
∼11.6 k	10%	0	0	15	15	0	0	10.87	10.87
∼15.8 k	–	0	88	185	185	0	89.39	385.97	385.97
∼15.8 k	10%	0	190	209	209	0	177.72	289.37	289.37
∼21.7 k	–	88	222	469	469	130.84	234.69	1147.25	1147.25
∼21.7 k	10%	74	406	526	526	97.36	607.79	1148.02	1148.02
∼61 k	–	1778	1089	1953	1926	4177.81	1118.63	5144.1	5090.34
∼61 k	10%	2149	1901	2221	2200	4517.07	4438.22	5457.13	5469.17

Increasing the number of jobs in traces (Figure 11) resulted in increasing the edge cost for all algorithms (Table 5(a)–(e)), which is also related to the number of jobs becoming available at a specific time (recall Figures 9 and 10). This is seen in Figure 11e–h. However, CECBS-R used fewer servers compared to other algorithms for scheduling and leveraged fewer servers less than 48 h. The more jobs become available for scheduling, the more servers are required; in particular, for privacy-sensitive jobs.

Table 5 shows that CECBS-R took advantage of public clouds and controlling the active servers to achieve cost efficiency. In contrast, all the baseline algorithms used the edge inefficiently, which led to higher costs that are also emerged from poor server management of the edge. Although the edge cost for CECBS-R is the lowest, the total cost of scheduling for ∼61 k jobs is the most among all algorithms. To investigate this, we present the number of QoS violations (i.e., number of missed deadlines) that happened for the traces in Table 6. This table presents that CECBS-R could reduce the number of QoS violations when the number of jobs increased. Although AsQ has the least total cost under privacy ratios, it has the highest violated QoS. BOS and RMS almost achieved the same cost after AsQ, with significantly fewer violations, but still higher than CECBS-R.

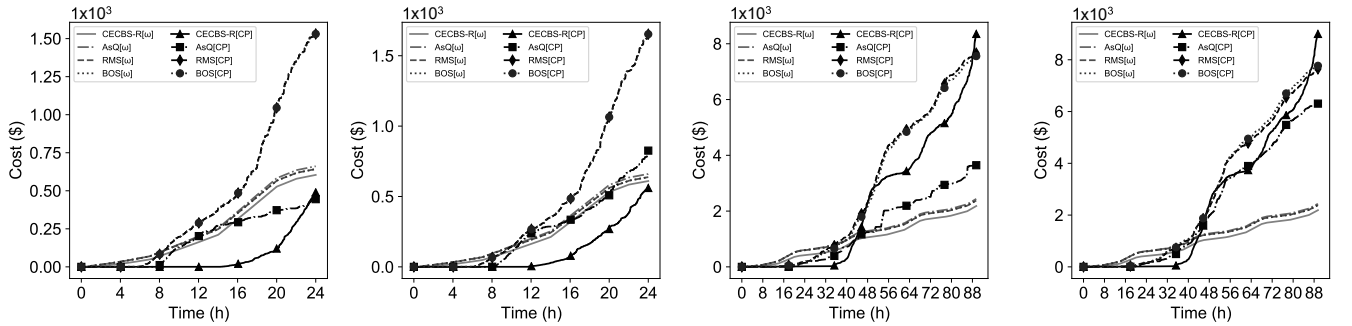


(**a**) ∼11.6 k without privacy (**b**) ∼11.6 k under 10% privacy (**c**) ∼15.8 k without privacy (**d**) ∼15.8 k under 10% privacy

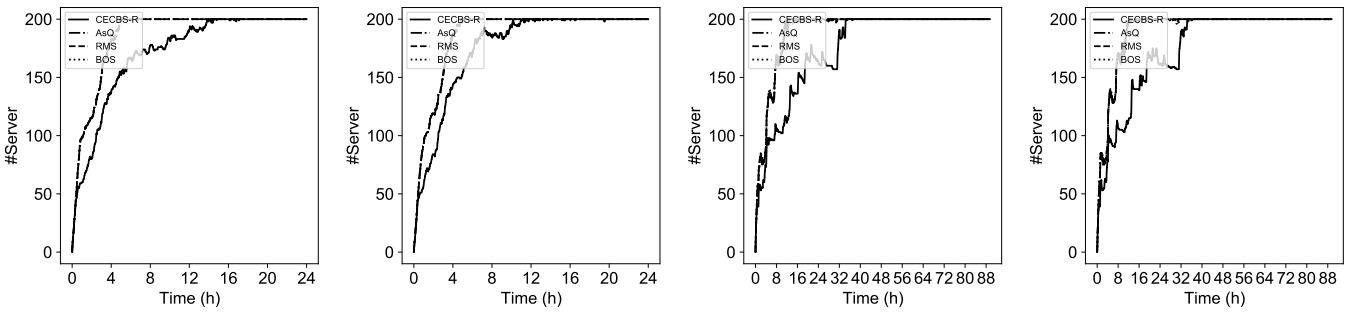


(**e**) 11.6 k without privacy (**f**) ∼11.6 k under 10% privacy (**g**) ∼15.8 k without privacy (**h**) ∼15.8 k under 10% privacy

Figure 10. The accumulated edge (ω) and public cloud (CP) cost followed by the edge server usage for ∼11.6 k and ∼15.8 k jobs.



(**a**) ∼21.7 k without privacy (**b**) ∼21.7 k under 10% privacy (**c**) ∼61 k without privacy (**d**) ∼61 k under 10% privacy



(**e**) ∼21.7 k without privacy (**f**) ∼21.7 k under 10% privacy (**g**) ∼61 k without privacy (**h**) ∼61 k under 10% privacy

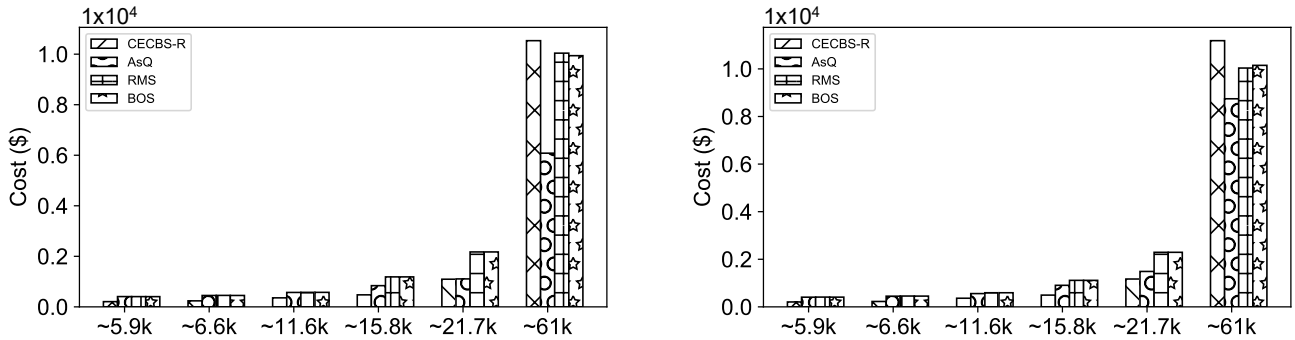
Figure 11. The accumulated edge (ω) and public cloud (CP) cost followed by the edge server usage for ∼21.7 k and ∼61 k jobs.

Table 6. #QoS violations.

Trace	Privacy	CECBS-R	AsQ	RMS	BOS
∼5.9 k	–	0	0	0	0
	10%	0	0	0	0
∼6.6 k	–	0	0	0	0
	10%	0	0	0	0
∼11.6 k	–	0	5	0	0
	10%	0	27	0	0
∼15.7 k	–	0	1316	0	0
	10%	0	1161	30	30
∼21.7 k	–	0	3848	0	0
	10%	10	3409	103	103
∼61 k	–	0	∼15.9 k	0	0
	10%	254	∼17.1 k	646	507

CECBS-R for ∼61 k trace relied more on public clouds (Table 5) for offloading purposes to comply with the QoS. In contrast, the heuristic-based CECBS-R scheduler led to a lower cost for the edge. The public cloud usage shows that CECBS-R considers the cost efficiency of resources on public clouds, while considering the active servers in the edge. Furthermore, AsQ, RMS, and BOS leveraged all the available servers inside the edge earlier than CECBS-R and achieved higher cost, while CECBS-R used fewer servers within the edge for half of a day. In fact, knowing runtimes without considering how cost-efficient it is to use either edge or public resources cannot lead to cost-saving.

Figure 12 shows the total cost when jobs have no privacy (Figure 12a) and under privacy ratios of 10% (Figure 12b) and illustrates that CECBS-R outperformed other algorithms. Even for the workload trace with the size of ∼61 k, CECBS-R could perform better than AsQ, because the number of QoS violations is the least for the CECBS-R shown in Table 6.



(**a**) Without privacy (**b**) Under 10% privacy ratio

Figure 12. The total cost across the edge-cloud environment without privacy and under 10% privacy.

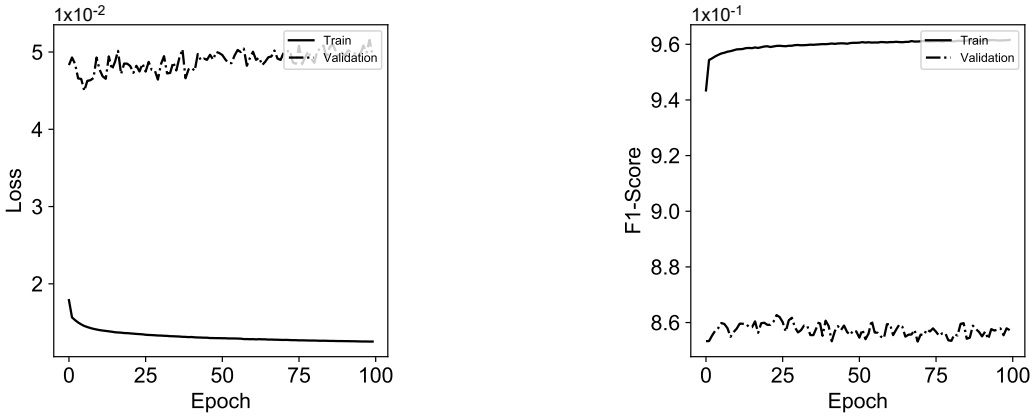
4.3. The Recommender Results

In this section, the result of CECBS-R recommender for an unseen and new Facebook workload trace (∼24.45 k jobs), as well as Karlsruhe Institute of Technology (KIT) System trace (∼44.5 k jobs) [40] across the edge-cloud environment is reported. The former is the real workload trace for 24 h while the latter is the traces collected over one year and a half in which the last four months of the trace are selected for this section. For recommendation, CECBS-R is trained by the previous scheduling results obtained from the CECBS-R scheduler under the given privacy ratios.

The CECBS-R recommender employs an LSTM neural network that consists of three layers. The input layer receives job submission time, length, required CPU cores and

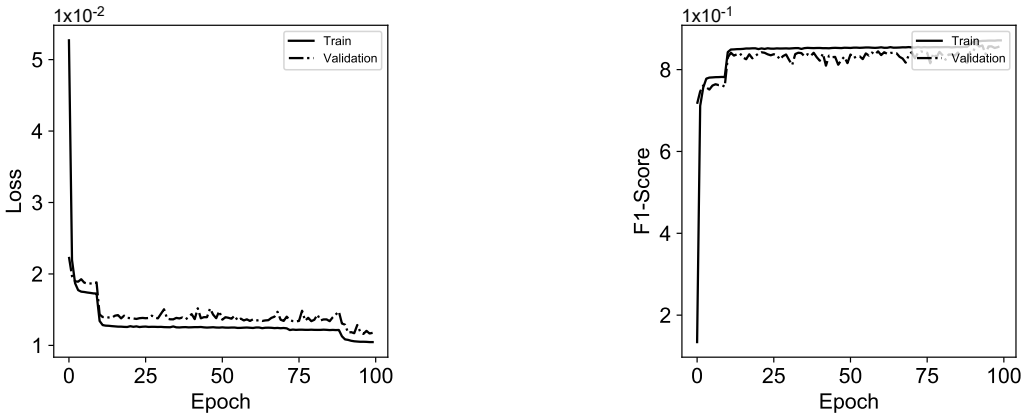
memory, privacy, deadline, the corresponding electricity rates category, and the edge utilization level at the time of job scheduling. The output is a one-hot vector associated with the job's cloud environment. Moreover, the training datasets are divided into two sets, with a ratio of 80:20 for training and validation, respectively.

Figure 13 shows the overall training error and the F1-score for the model validation considering the previous job executed information. The loss and F1-score for the resource recommender with respect to the collected job information show that the model could learn a diverse job resource allocation, as the training error is still low. This figure is followed by the re-trained MLP neural network (Figure 14) for runtime estimation taking into account the executed job information across the edge-cloud environment. Due to the job diversities within the workloads, the F1-score for the validation set is almost ∼85%, which could still be leveraged for the recommendation. The model is retrained with the new collected job information, meaning that the designed model should have to be re-adjusted, hence, it is expected to have an increment in the model error, and a lower F1-score for the runtime estimation. Figure 15 shows that considering the recommendation eventually led to cost efficiency with the least QoS violations. Although AsQ could achieve the least cost for the trace ∼24.4 k without privacy, the CECBS-R could guarantee the QoS of jobs slightly at a higher cost than AsQ.



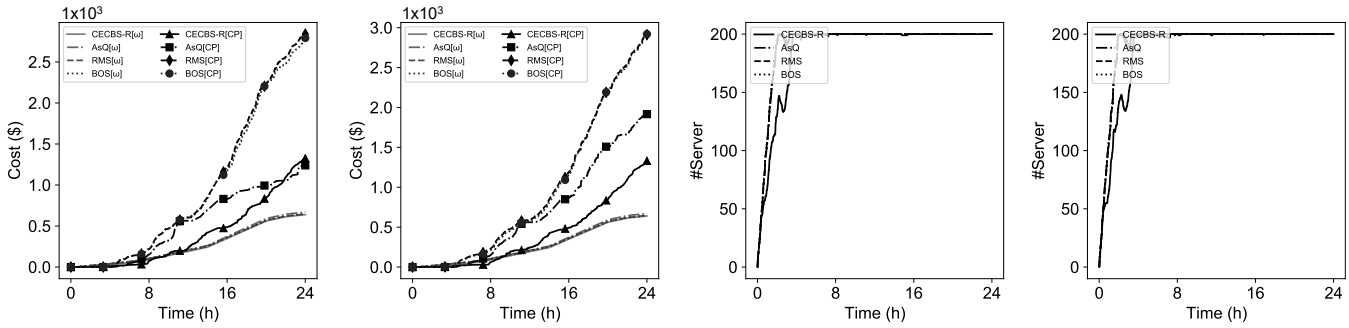
(**a**) Training error (**b**) F1-score

Figure 13. The loss and F1-score over 100 epochs for the recommender.



(**a**) Training error (**b**) F1-score

Figure 14. The loss and F1-score over 100 epochs for the runtime estimation model.



(**a**) Without privacy (**b**) Under 10% privacy (**c**) Without privacy (**d**) Under 10% privacy

Figure 15. The accumulated edge (ω) and public cloud (CP) cost followed by the edge server usage for ∼24.4 k jobs.

Figure 15c,d shows how CECBS-R utilized edge servers better than the other algorithms. For the trace contains ∼24.45 k jobs, AsQ, RMS, and BOS almost used all the servers in the edge towards the end of execution, however, nearly a quarter of the time CECBS-R used fewer resources than the other algorithms. This resulted in different edge costs for the baseline algorithms and CECBS-R. While CECBS-R relied on the recommendations, on average, it achieved a lower edge cost than the other algorithms when jobs have no privacy and under a 10% privacy ratio. Although CECBS-R sent off more jobs to the edge (Table 5(a)), it effectively used the resources which led to cost efficiency because of being helped by the PSO algorithm. The usage of edge servers could increase the chance of relying on the public cloud for job execution (Figure 15a,b). However, for CECBS-R, the heuristic fitness function Equation (17) recommended that jobs were executed in the edge aimed at increasing resource utilization, while achieving a lower cost. Higher server utilization would increase the wattage usage with respect to Equation (4); consequently, it would increase the usage cost, but at the same time, underutilized servers would also hurt the cost-efficiency, as idle servers in a data center consume 50% of peak power [41,42].

Figure 16 shows that CECBS-R assisted by neural networks achieved the lowest cost for longer traces and comparing to other algorithms when privacy is in effect, CECBS-R relied on the public resources (see Table 7). In contrast, AsQ, RMS, and BOS relied on edge resources for job execution and hardly ever used public clouds. Table 7a (and Table 7(b)–(e)) show that after CECBS-R, RMS almost sent off more jobs to the edge, and AsQ offloaded more jobs to the public clouds for the trace ∼24.4 k. CECBS-R for higher privacy ratios steadily tended to employ both edge and public resources, in particular, relying more on the edge servers. It should be noted that even with the availability of runtimes in advance for baseline algorithms, the scheduler/recommender plays an important part in making judicious scheduling decisions, as baseline algorithms could not achieve cost efficiency for any of the traces.

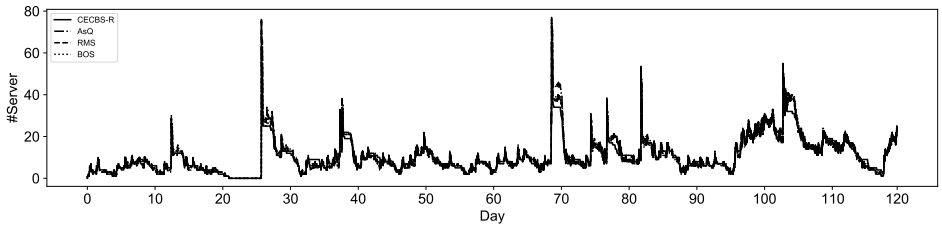
Table 7 shows that CECBS-R outperformed the baseline algorithms, as CECBS-R could learn what resources would be more cost-efficient across the edge-cloud environment. It can be seen that CECBS-R leveraged public resources (Table 7(a)–(e)) for the trace ∼44.5 k, as it was cost efficient to use resources there. CECBS-R could perfectly be aligned with the workload pattern as comparing Figure 15c,d with Figure 16 shows that the usage of the edge could be close to the baseline algorithm but in a more efficient way. Less spikes can be seen in Figure 16 for CEBCS-R, e.g., days ∼27, ∼40, ∼70, and ∼105. Recall the fact that leaving edge resource underutilized (or idle) could lead to increasing the utility cost which can be seen in Table 7. Consequently, cloud bursting recommendations based on LSTM significantly reduced the total cost across the edge-cloud environment. These results show how CECBS-R could recommend better judicious decisions based on the learnt workload characteristics and their resource requirements for achieving cost savings that are aligned with Figures 15, 17 and 18. Moreover, Table 8 shows the number of QoS violations for CECBS-R and the other algorithms as for ∼44.5 trace CECBS-R slightly violated more jobs' QoS.

Table 7. Edge-cloud job proportion and its cost under volatile (ω^v) electricity rates.

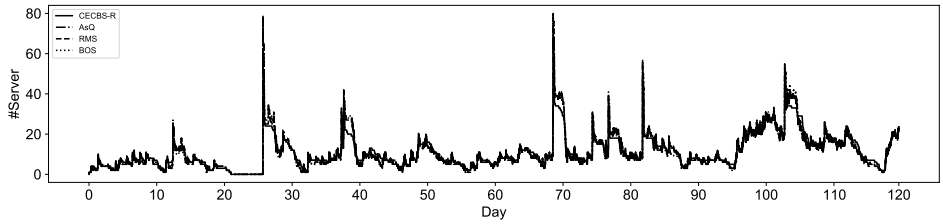
(a) Edge									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼24.4 k	–	20,872	18,826	19,559	19,495	638.95	669.75	645.64	647.01
∼24.4 k	10%	20,905	16,594	23,144	17,939	639.6	670.77	647.71	645.56
∼44.5 k	–	42,853	44,466	44,466	44,466	4994.96	40,550.98	40,572.28	40,527.96
∼44.5 k	10%	42,868	44,466	44,466	44,466	5334.43	40,589.12	40,600.66	40,356.52
(b) AWS									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼22.4 k	–	2434	1084	299	253	48.8	101.56	73.14	52.62
∼22.4 k	10%	2289	1456	14	563	48.46	87.62	102.05	90.66
∼44.5 k	–	1148	0	0	0	73.55	0	0	0
∼44.5 k	10%	1150	0	0	0	87.83	0	0	0
(c) GCE									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼24.4 k	–	52	75	106	95	13.97	65.06	151.66	129.49
∼24.4 k	10%	55	127	23	114	13.66	111.02	171.95	179.75
∼44.5 k	–	65	0	0	0	61.89	0	0	0
∼44.5 k	10%	60	0	0	0	17.9	0	0	0
(d) MS									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼24.4 k	–	570	4128	3592	3706	672.12	568.24	662.63	676.89
∼24.4 k	10%	629	5481	976	4929	633.12	444.3	553.46	642.21
∼44.5 k	–	173	0	0	0	181.68	0	0	0
∼44.5 k	10%	162	0	0	0	198.84	0	0	0
(e) AC									
Trace	**Privacy**	**Job Proportion**				**Cost ($)**			
		CECBS-R	AsQ	RMS	BOS	CECBS-R	AsQ	RMS	BOS
∼24.4 k	–	515	330	887	894	591.59	504.15	1951.81	1932.87
∼24.4 k	10%	565	785	286	898	635.02	1273.6	2085.9	2006.55
∼44.5 k	–	227	0	0	0	307.9	0	0	0
∼44.5 k	10%	226	0	0	0	396.51	0	0	0

Table 8. #QoS violations.

Trace	Privacy	CECBS-R	AsQ	RMS	BOS
∼24.4 k	–	0	4235	0	0
	10%	63	4886	146	137
∼44.5 k	–	10	0	0	0
	10%	8	0	0	0

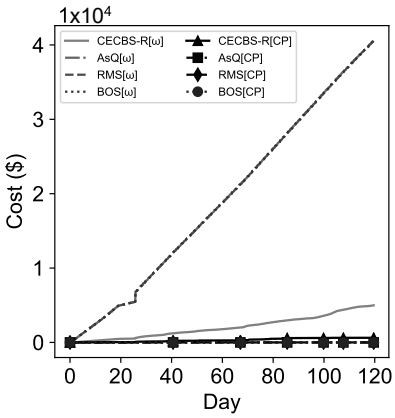


(**a**) Without privacy

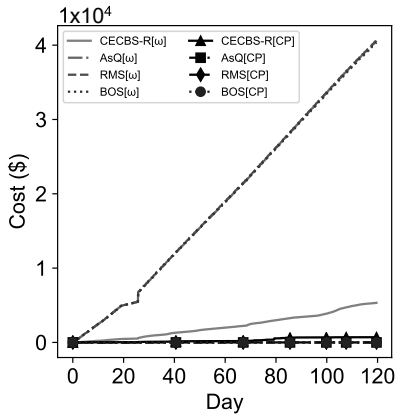


(**b**) Under 10% privacy

Figure 16. The edge server usage for ∼44.5 k jobs during 3 months.

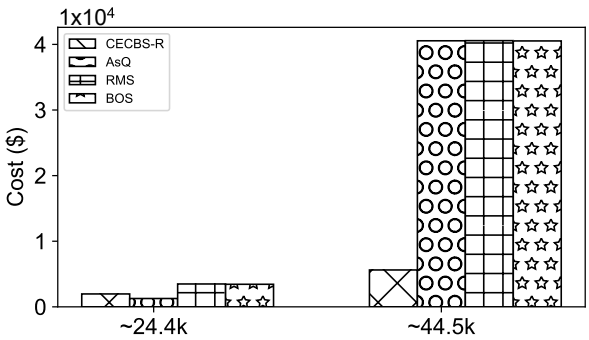


(**a**) Without privacy

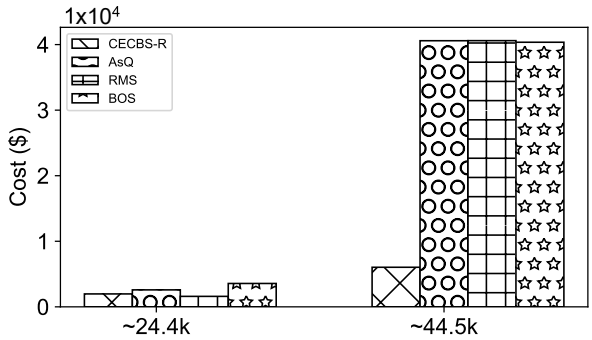


(**b**) Under 10% privacy

Figure 17. The accumulated edge (ω) and public cloud (CP) cost for ∼44.5 k jobs.



(**a**) Without privacy



(**b**) Under 10% privacy

Figure 18. The total cost across the edge-cloud environment without privacy and under 10% privacy.

5. Related Work

Cloud bursting has been extensively studied [12,22,43–51] and studies [5–8,52–58] pay attention to scheduling for improving cost and/or performance. However, a majority of studies fail to notice the impact of the edge (i.e., in situ resources) utility cost in cost efficiency, while preserving privacy. Moreover, due to the presence of workload runtime

uncertainties, the cost-efficiency of cloud bursting becomes more challenging, but can be addressed through applying machine learning techniques for resource selection or recommendation [15,17].

Calheiros' work [5] is about deadline-constraint job scheduling, in which job queues and resource queues are used to allocate resources for deadline-sensitive or deadline-insensitive jobs. Cost optimization can be found in [7,8,44,49,51]. In these works, performance [7,44,49] and minimizing the public cloud cost [8,51] are studied considering the improvement of local resource utilization. Chen et al. [52] present how cost evaluation can be effective for choosing off-premise resources. Bossche et al., [54,56] show the potential cost-efficient cloud environment selection for deadline-constraint jobs through cost evaluation. He also, in another study [6], developed a four-strategy algorithm that cost-effectively manages submitted jobs to the edge to be executed across a hybrid cloud. The developed online hybrid scheduler leverages two offloading policies for resource selection for the public cloud, as well as two job selection strategies for incoming jobs at the edge when they are deadline-sensitive.

As CECBS-R is assisted by ML techniques and employs edge-cloud resources, it falls in the context of cyber-physical systems (CPSs) [59], where the risk assessment (i.e., preserving privacy, e.g., user data) is controlled through the proper allocation of resources across the edge while meeting incoming jobs' resource requirements [60]. Moreover, due to the complexities of cloud bursting for software products, Acs et al. [12] offer a nested virtualization technique to seamlessly overcome the issues and the cloud differences. The usage of a data locality for iterative MapReduce applications through cloud bursting [43], a heuristic approach for task sequencing and task scheduling for minimizing bags-of-task completion, and through binary linear programming [55], includes research works in cost-efficient resource allocation. Furthermore, Chunlin et al. [47] address resource-intensive mobile application scheduling, which leverages service-level agreement (SLA) to determine the share of both edge and public clouds resources to be allocated to a mobile user.

Data-driven approaches have also gained attention for resource allocation [46,48,50,61]. In [46], a support vector machine (SVM) is used to profile and anticipate user activities for resource management in a multimedia cloud environment, and [48] proposes a framework that relies on the processing time estimations to feasibly allocate resources to jobs. Champati et al. [50] discuss an online runtime estimation-based algorithm to efficiently minimize makespan in the edge while keeping the offloading minimum, and Sonmez et al. [61] presented a two-stage machine learning-based load orchestrator for task allocation in edge/cloud servers.

Optimal resource allocation which leads to cost-efficiency are studied through a multi-dimension knapsack problem [22] and Pareto-optimality [53,62]. Charrada and Tata [57] propose a cost calculation strategy for resource selection based on procedure-based approaches, while bursting workloads to the public cloud.

Although these studies address cloud bursting cost efficiency and/or dealing with workload surges, neither workload privacy nor the utility cost is considered. Hence, this paper not only considers the edge utility cost and workload privacy, but also takes advantage of machine learning techniques to facilitate cloud bursting for achieving better cost efficiency. Therefore, this paper is distinguishable from the studies above, as the edge utility cost under various electricity rates is taken into account for job scheduling under privacy constraints. Machine learning techniques are used to deal with runtime uncertainties and assist cloud bursting for choosing the cost-efficient cloud/edge environment, which is done by learning job characteristics and resource requirements.

6. Conclusions

This paper has studied the cost efficiency of AI-assisted scheduling in the edge-cloud environment with the development of CECBS-R as a dual-objective framework for scheduling and recommendation. CECBS-R incorporated machine learning techniques such as MLP, LSTM, and LSTM neural networks that were responsible for learning, training,

and recommendation in the edge-cloud environment. CECBS-R not only preserved privacy, but also considered the edge utility cost plus public cloud billing cycles for scheduling for cost-efficient scheduling. Moreover, CECBS-R by profiling jobs assisted scheduling through resource recommendations as well as runtime estimation. CECBS-R is trained with the scheduling outputs of Facebook and grid workload traces as real workload traces. CECBS-R has shown that considering the advantages of the edge-cloud environment can lead to cost efficiency, and a resource recommender could be in charge of the cost-efficient resource allocation. Results under different scenarios compared to the state-of-the-art algorithms have confirmed our claims. For our future work, resource-constrained devices in addition to the heterogeneous resources across the edge-cloud environment will be considered to provide more insights into their contribution to resource allocation.

Author Contributions: Formal analysis, A.P., Y.C.L., T.H. and K.A.; Methodology, A.P.; Project administration, Y.C.L.; Supervision, Y.C.L.; Validation, Y.C.L., T.H. and K.A.; Writing—original draft, A.P.; Writing—review and editing, A.P. and Y.C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

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

References

1. Amazon EC2 Pricing. 2020. Available online: https://aws.amazon.com/ec2/pricing/ (accessed on 21 May 2021).
2. Microsoft Azure Pricing. 2020. Available online: https://azure.microsoft.com/en-au/pricing/ (accessed on 21 May 2021).
3. Google Compute Engine. 2020. Available online: https://cloud.google.com/compute/pricing (accessed on 21 May 2021).
4. Alibaba Cloud. 2020. Available online: https://www.alibabacloud.com (accessed on 21 May 2021).
5. Calheiros, R.N.; Buyya, R. Cost-Effective Provisioning and Scheduling of Deadline-Constrained Applications in Hybrid Clouds. In *Web Information Systems Engineering—WISE 2012*; Wang, X.S., Cruz, I., Delis, A., Huang, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 171–184.
6. Den Bossche, R.V.; Vanmechelen, K.; Broeckhove, J. Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. *Future Gener. Comput. Syst.* **2013**, *29*, 973–985. [CrossRef]
7. Bittencourt, L.F.; Senna, C.R.; Madeira, E.R. Scheduling service workflows for cost optimization in hybrid clouds. In Proceedings of the 2010 International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2–6 November 2010; pp. 394–397.
8. Lee, Y.; Lian, B. Cloud Bursting scheduler for cost efficiency. In Proceedings of the 10th IEEE International Conference on Cloud Computing, Honolulu, HI, USA, 25–30 June 2017; pp. 774–777.
9. Clemente-Castelló, F.J.; Mayo, R.; Fernández, J.C. Cost Model and Analysis of Iterative MapReduce Applications for Hybrid Cloud Bursting. In Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 14–17 May 2017; pp. 858–864.
10. Chu, H.; Simmhan, Y. Cost-Efficient and Resilient Job Life-Cycle Management on Hybrid Clouds. In Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, 19–23 May 2014; pp. 327–336.
11. Hoseinyfarahabady, M.R.; Samani, H.R.D.; Leslie, L.M.; Lee, Y.C.; Zomaya, A.Y. Handling Uncertainty: Pareto-Efficient BoT Scheduling on Hybrid Clouds. In Proceedings of the 2013 42nd International Conference on Parallel Processing, Lyon, France, 1–4 October 2013; pp. 419–428.
12. Acs, S.; Kozlovszky, M.; Kacsuk, P. A novel cloud bursting technique. In Proceedings of the 2014 IEEE 9th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 15–17 May 2014; pp. 135–138.
13. Ilyushkin, A.; Epema, D. The Impact of Task Runtime Estimate Accuracy on Scheduling Workloads of Workflows. In Proceedings of the 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Washington, DC, USA, 1–4 May 2018; pp. 331–341.
14. Soysal, M.; Berghoff, M.; Klusáček, D.; Streit, A. On the quality of wall time estimates for resource allocation prediction. In Proceedings of the 48th International Conference on Parallel Processing: Workshops, Kyoto, Japan, 5–8 August 2019; pp. 1–8.
15. Cortez, E.; Bonde, A.; Muzio, A.; Russinovich, M.; Fontoura, M.; Bianchini, R. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28–31 October 2017; ACM: New York, NY, USA, 2017; pp. 153–167.
16. Microsoft Azure Traces. 2019. Available online: https://github.com/Azure/AzurePublicDataset (accessed on 21 May 2021).

17. Yadwadkar, N.J.; Hariharan, B.; Gonzalez, J.E.; Smith, B.; Katz, R.H. Selecting the Best VM Across Multiple Public Clouds: A Data-driven Performance Modeling Approach. In Proceedings of the 2017 Symposium on Cloud Computing, Santa Clara, CA, USA, 24–27 September 2017; ACM: New York, NY, USA, 2017; pp. 452–465.
18. Duc, T.L.; Leiva, R.G.; Casari, P.; Östberg, P.O. Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey. *ACM Comput. Surv.* **2019**, *52*. [CrossRef]
19. Pasdar, A.; Lee, Y.C.; Almi'ani, K. Toward Cost Efficient Cloud Bursting. In *Service-Oriented Computing*; Yangui, S., Bouassida Rodriguez, I., Drira, K., Tari, Z., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 299–313.
20. Facebook Workload Traces. 2019. Available online: https://github.com/SWIMProjectUCB/SWIM/wiki/Workloads-repository (accessed on 21 May 2021).
21. Grid Workload. 2020. Available online: http://gwa.ewi.tudelft.nl/datasets/gwa-t-11-lcg (accessed on 21 May 2021).
22. Wang, W.J.; Chang, Y.S.; Lo, W.T.; Lee, Y.K. Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments. *J. Supercomput.* **2013**, *66*, 783–811. [CrossRef]
23. Power Supply Calculator. 2019. Available online: https://outervision.com/power-supply-calculator (accessed on 21 May 2021).
24. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.* **2012**, *28*, 755–768. [CrossRef]
25. Lim, S.H.; Sharma, B.; Tak, B.C.; Das, C.R. A dynamic energy management scheme for multi-tier data centers. In Proceedings of the (IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software, Austin, TX, USA, 10–12 April 2011; IEEE: New York, NU, USA, 2011; pp. 257–266.
26. Fan, X.; Weber, W.D.; Barroso, L.A. Power provisioning for a warehouse-sized computer. *ACM SIGARCH Comput. Archit. News* **2007**, *35*, 13–23. [CrossRef]
27. Reiss, C. Towards understanding heterogeneous clouds at scale: Google trace analysis. In Proceedings of the Third ACM Symposium on Cloud Computing, San Jose, CA, USA, 23–25 October 2012.
28. Energy Australia. 2019. Available online: https://www.energyaustralia.com.au/business/electricity-and-gas/small-business/plans (accessed on 21 May 2021).
29. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 21 May 2021).
30. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
31. Mao, M.; Humphrey, M. A performance study on the vm startup time in the cloud. In Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, 24–29 June 2012; pp. 423–430.
32. Microsoft Azure Deployment Time. 2019. Available online: https://feedback.azure.com/forums/216843-virtual-machines/suggestions/5715040-accelerate-vm-startup-time (accessed on 21 May 2021).
33. Google Compute Engine Deployment Time. 2019. Available online: https://medium.com/google-cloud/understanding-and-profiling-gce-cold-boot-time-32c209fe86ab (accessed on 21 May 2021).
34. Sizing Your IBM® Cloud Private Cluster. 2019. Available online: https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.1.2/installing/plan_capacity.html (accessed on 21 May 2021).
35. Energy Australia: Energy Price Fact Sheet NSW Business (Elec-Tricity). 2017. Available online: https://secure.energyaustralia.com.au/EnergyPriceFactSheets/Docs/EPFSIE_B_N_BBAS_EA_6_02-01-2017.pdf (accessed on 21 May 2021).
36. Keras Python Deep Learning API. 2020. Available online: https://keras.io (accessed on 21 May 2021).
37. Bitbrains VMs Performance. 2019. Available online: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains (accessed on 21 May 2021).
38. VMware, Inc. 2019. Available online: https://www.vmware.com/au.html (accessed on 21 May 2021).
39. Grid Workload. 2020. Available online: http://gwa.ewi.tudelft.nl/datasets/gwa-t-1-das2 (accessed on 21 May 2021).
40. Karlsruhe Institute of Technology Workload Traces. 2019. Available online: https://www.cs.huji.ac.il/labs/parallel/workload/l_kit_fh2/index.html (accessed on 21 May 2021).
41. Greenberg, A.; Hamilton, J.; Maltz, D.A.; Patel, P. The Cost of a Cloud: Research Problems in Data Center Networks. *SIGCOMM Comput. Commun. Rev.* **2008**, *39*, 68–73. [CrossRef]
42. Barroso, L.A.; Hölzle, U. The Case for Energy-Proportional Computing. *Computer* **2007**, *40*, 33–37. [CrossRef]
43. Clemente-Castello, F.J.; Nicolae, B.; Rafique, M.M.; Mayo, R.; Fernandez, J.C. Evaluation of Data Locality Strategies for Hybrid Cloud Bursting of Iterative MapReduce. In Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 14–17 May 2017; pp. 181–185.
44. Loreti, D.; Ciampolini, A. A Hybrid Cloud Infrastructure for Big Data Applications. In Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, New York, NY, USA, 24–26 August 2015; pp. 1713–1718.
45. Zhang, Y.; Sun, J.; Wu, Z. An Heuristic for Bag-of-Tasks Scheduling Problems with Resource Demands and Budget Constraints to Minimize Makespan on Hybrid Clouds. In Proceedings of the 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD), Shanghai, China, 13–16 August 2017; pp. 39–44.
46. Daniel, D.; Raviraj, P. Distributed Hybrid Cloud for Profit Driven Content Provisioning Using User Requirements and Content Popularity. *Cluster Comput.* **2017**, *20*, 525–538. [CrossRef]

47. Chunlin, L.; Jianhang, T.; Youlong, L. Distributed QoS-aware scheduling optimization for resource-intensive mobile application in hybrid cloud. *Cluster Comput.* **2018**, *21*, 1331–1348. [CrossRef]
48. Zhu, J.; Li, X.; Ruiz, R.; Xu, X. Scheduling Stochastic Multi-Stage Jobs to Elastic Hybrid Cloud Resources. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 1401–1415. [CrossRef]
49. Zuo, L.; Shu, L.; Dong, S.; Chen, Y.; Yan, L. A Multi-Objective Hybrid Cloud Resource Scheduling Method Based on Deadline and Cost Constraints. *IEEE Access* **2017**, *5*, 22067–22080. [CrossRef]
50. Champati, J.P.; Liang, B. One-restart algorithm for scheduling and offloading in a hybrid cloud. In Proceedings of the 2015 IEEE 23rd International Symposium on Quality of Service (IWQoS), Portland, OR, USA, 15–16 June 2015; pp. 31–40.
51. Guo, T.; Sharma, U.; Wood, T.; Sahu, S.; Shenoy, P. Seagull: Intelligent cloud bursting for enterprise applications. In Proceedings of the 2012 {USENIX} Annual Technical Conference ({USENIX}{ATC} 12), Boston, MA, USA, 13–15 June 2012; pp. 361–366.
52. Chen, Y.; Sion, R. To Cloud or Not to Cloud? Musings on Costs and Viability. In Proceedings of the 2nd ACM Symposium on Cloud Computing, Cascais, Portugal, 23–25 October 2011; ACM: New York, NY, USA, 2011; p. 29.
53. Farahabady, M.R.H.; Lee, Y.C.; Zomaya, A.Y. Pareto-Optimal Cloud Bursting. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 2670–2682.
54. Den Bossche, R.V.; Vanmechelen, K.; Broeckhove, J. Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads. In Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, Miami, FL, USA, 5–10 July 2010; pp. 228–235.
55. Abdi, S.; PourKarimi, L.; Ahmadi, M.; Zargari, F. Cost Minimization for Deadline-constrained Bag-of-tasks Applications in Federated Hybrid Clouds. *Future Gener. Comput. Syst.* **2017**, *71*, 113–128. [CrossRef]
56. Den Bossche, R.V.; Vanmechelen, K.; Broeckhove, J. Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds. In Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, Athens, Greece, 29 November–1 December 2011; pp. 320–327.
57. Charrada, F.B.; Tata, S. An Efficient Algorithm for the Bursting of Service-Based Applications in Hybrid Clouds. *IEEE Trans. Serv. Comput.* **2016**, *9*, 357–367. [CrossRef]
58. Li, C.; Tang, J.; Luo, Y. Towards operational cost minimization for cloud bursting with deadline constraints in hybrid clouds. *Cluster Comput.* **2018**, *21*, 2013–2029. [CrossRef]
59. Radanliev, P.; De Roure, D.; Van Kleek, M.; Santos, O.; Ani, U. Artificial intelligence in cyber physical systems. *AI Soc.* **2020**, 1–14. [CrossRef]
60. Radanliev, P.; De Roure, D.; Walton, R.; Van Kleek, M.; Montalvo, R.M.; Santos, O.; Burnap, P.; Anthi, E.; others. Artificial intelligence and machine learning in dynamic cyber risk analytics at the edge. *SN Appl. Sci.* **2020**, *2*, 1–8. [CrossRef]
61. Sonmez, C.; Tunca, C.; Ozgovde, A.; Ersoy, C. Machine Learning-Based Workload Orchestrator for Vehicular Edge Computing. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 2239–2251. [CrossRef]
62. Ben-Yehuda, O.A.; Schuster, A.; Sharov, A.; Silberstein, M.; Iosup, A. ExPERT: Pareto-Efficient Task Replication on Grids and a Cloud. In Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium, Shanghai, China, 21–25 May 2012; pp. 167–178.