Eris: An Online Auction for Scheduling Unbiased Distributed Learning Over Edge Networks

Jinlong Pang, Ziyi Han, Ruiting Zhou, *Member, IEEE*, Renli Zhang, John C.S. Lui, *Fellow, IEEE*, and Hao Chen

***Abstract*—The emergence of edge intelligence has made smart IoT services (e.g., video/audio surveillance, autonomous driving and smart city) a reality. To ensure the quality of service, edge service providers train unbiased models of distributed machine learning jobs over the local datasets collected by edge networks, and usually adopt the parameter server (PS) architecture. However, the training of *unbiased distributed learning* (UDL) depends on geo-distributed data and edge resources, bringing a new challenge for service providers: how to effectively schedule and price UDL jobs such that the long-term system utility (i.e., social welfare) can be maximized. In this paper, we propose an online auction-based scheduling algorithm *Eris*, which determines the data workload, the number and the placement of concurrent workers and PSs for each arriving UDL job, and dynamically prices limited edge resources based on current resource consumption. *Eris* applies a primal-dual framework which calls an efficient dual subroutine to schedule UDL jobs, achieving a good competitive ratio and pseudo-polynomial time complexity. To evaluate the effectiveness of *Eris*, we implement both a testbed and a large-scaled simulator. The results demonstrate that *Eris* outperforms and achieves up to 44% more social welfare compared to state-of-the-art algorithms in today's cloud system.**

***Index Terms*—Distributed machine learning, auction, online scheduling.**

I. Introduction

THE combination of edge computing and artificial intelligence (AI) introduced a new promising paradigm, namely "edge intelligence" [1]. Instead of relying on the cloud entirely, edge intelligence utilizes massive data collected in IoT devices and edge servers to provide real-time services for smart IoT applications, e.g., intelligent video/audio surveillance, smart city and autonomous driving. To ensure IoT service's quality, it is critical to train an *unbiased* model[1] with the data collected by edge sites [3], [4], where an edge site can be an edge server attached to an access point (e.g., WiFi hotspots or base station), an edge cloudlet, or even a network gateway. For example, to deploy smart traffic management system, real-time road traffic data collected by installed cameras needs to be uploaded to nearby edge cloudlets for processing [5], [6], [7]. The collected data from different edge cloudlets are used to train an unbiased model which can work in all roads to mange the traffic. In this work, we define this type of learning jobs as the *unbiased distributed learning* (UDL) jobs. Unlike traditional distributed learning, UDL jobs source their training data from edge sites, which offer both computational resources and the necessary training data. Note that one common approach to training UDL jobs is to adopt *data parallelism* with the "parameter server (PS)" framework at the edge [3]. In each training iteration, the workers deployed for one UDL job (either implemented on virtual machines (VM) or cloud containers) compute gradients based on input datasets, and then send gradients to PSs. The PSs update model parameters by averaging gradients and then push parameters to all workers.

For the service provider, who owns the training data and edge resources, there exists two unique concerns need to be addressed when training UDL jobs: *technical* and *economic*. *First*, on the technical side, given heterogeneous resources (especially scarce and expensive bandwidth) and data volumes in different edge sites, it is challenging to decide the training data workload in each site, and deploy the matched number of workers, such that the average job completion time is minimized and total bandwidth consumption is as small as possible. Furthermore, due to the frequent communication between workers and PSs, the placement of PSs also affects the inter-site bandwidth consumption and the job completion time if workers and PSs are not deployed on the same site. Therefore, the training of UDL jobs needs to jointly optimize bandwidth consumption and minimize the average job completion time. *Second*, on the economic side, it is non-trivial to price limited edge resources upon the arrival of each UDL job without any future information such that the system's utility in the long run (i.e., social welfare) can be maximized. One tailored pricing scheme is needed to charge UDL jobs, considering both specific features of the edge (i.e., resource scarcity) and UDL jobs (i.e., geo-distributed data).

Manuscript received 4 March 2023; revised 26 October 2023; accepted 6 November 2023. Date of publication 29 November 2023; date of current version 7 May 2024. This work was supported in part by the NSFC under Grants 62072344, U20A20177, and 62232004. Recommended for acceptance by T. Shu. *(Corresponding author: Ruiting Zhou.)*

Jinlong Pang, Ziyi Han, and Renli Zhang are with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: jinlongpang@whu.edu.cn; ziyihan@whu.edu.cn; zhang_rl@whu.edu.cn).

Ruiting Zhou is with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: ruitingzhou@seu.edu.cn).

John C.S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

Hao Chen is with Huawei Technologies Company, Ltd., Nanjing 210012, China (e-mail: philips.chenhao@huawei.com).

Digital Object Identifier 10.1109/TMC.2023.3333368

[1]An *unbiased* model represents a model with high applicability that can be used in all sites to provide effective inference [2].

1536-1233 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

There have been some efforts to address the above two concerns. For the technical side, commonly adopted schedulers in cloud platforms [8], [9] follow intuitive strategies, e.g., First In First Out (FIFO) or Dominant Resource Fairness Scheduling [10]. However, for these cloud schedulers, providing the fixed configuration, i.e., the number of workers/PSs and the training time, is mandatory, which leads to inflexibility of resource scheduling [11]. For the edge network, recent work mainly focuses on task offloading problems [12], [13], [14]. For the economic side, fixed pricing, where charging a fixed unit price for the utilized resources based on jobs' training time, is commonly adopted [8], [15]. However, fixed pricing fails to capture the changing supply-demand relationship in the market. Consequently, the case of overpricing or underpricing routinely happens, which will jeopardize the service operator's profit as well as the utility of the entire system. Therefore, fixed pricing is not suitable for UDL jobs due to the uncertainty of job's training time, which depends on the job placement. In this regard, auction is a natural pricing approach which can automatically set the right price and further allocate resources to users who value them most. Auction can further enable the users to bid for any combination of resources or configurations. Besides, auction-based mechanisms can effectively guarantee truthfulness and individual rationality. A number of auction-based mechanisms in different perspectives [16], [17], [18], [19], [20], [21] have been proposed, which focus both on pricing mechanism and resource allocation. However, these auction-based pricing mechanisms do not compatible with ML jobs with uncertain training time. More details are presented in Section II.

Herein, we develop an auction-based framework *Eris* for UDL jobs to tackle both the technical and economic challenges. *Eris* dynamically schedules each UDL job upon its arrival under the bandwidth consumption constraint to maximize social welfare, which is equivalent to minimizing the average job completion time (because a UDL job's utility is non-increasing with its job completion time). On the other hand, *Eris* applies auction mechanisms to automatically price limited edge resources. *Eris* mainly consists of two modules: 1) an online scheduling algorithm that decides the amount of training data, the execution time window, the number and the placement of concurrent workers and PSs at each time slot upon the arrival of each UDL job; and 2) one well-designed pricing scheme that adjusts resource price (including bandwidth) over time based on the current resource usage to prevent users from submitting jobs and enable the operator to avoid resource starvation. To the best of our knowledge, this work is the *first formal study of dynamic scheduling and pricing for UDL jobs at edge networks*. Specifically, our contributions are:

▷ Through capturing UDL jobs' distinct features, we model the social welfare maximization problem into a NP-hard mixed integer nonlinear program (MINLP). To tackle this problem, we first reformulate this problem into a typical knapsack-type problem by leveraging exponential number of variables.

▷ An effective auction-based scheduling algorithm *Eris* is presented. First, *Eris* estimates the resources cost for each job by using the dual variables which can be interpreted as unit resource prices. The price function reflects the market rules and is computed based on the resource consumption. Given the resource prices, *Eris* then calculates the optimal schedule for each accepted job via a dynamic programming algorithm.

▷ We rigorously prove the properties of *Eris*, including: i) correctness, ii) truthfulness, iii) polynomial running time, and iv) competitive ratio.

▷ To evaluate the superiority of *Eris*, extensive testbed experiments and large-scaled simulations are conducted. We implemented the testbed based on MXNet Framework with Kubernetes. We obtain promising results in Section VI: i) testbed experiments show that *Eris* achieves a near-optimal social welfare with low competitive ratio (< 1.5), which is much smaller than the theoretical bound; ii) *Eris* improves 44%, 50%, 20% social welfare compared to *Tiresias* [11], *AntMan* [22] and *Liquid* [23], respectively; iii) in large-scaled simulations, *Eris* always outperforms three respective baselines regardless of the input scales and price functions, which coincides with the observation of tested experiments.

The rest of this paper is listed as follows. First, Section II provides more details about related work. Then, Section III shows our system model. We introduce our approach and discuss its properties in Sections IV and V, respectively. Section VI evaluates the effectiveness of the proposed approach. The conclusion is presented in Section VII.

II. Related Work

Scheduling and placing in Edge/Cloud: In the energy-constrained scenario, to minimize the latency, Saleem et al. [13] investigate the mobility-aware task scheduling problem. Alameddine et al. [14] attempt to maximize the number of admitted latency-sensitive tasks by optimizing task offloading and scheduling. However, the aforementioned work usually allocate resources for tasks by leveraging virtual machine (VM) or cloud container rather than workers/PSs using in the typical "parameter server (PS)" architecture. Therefore, these work are not applicable for smart IoT applications. In cloud networks, how to make optimal decisions for job scheduling and placing has been explored over the last decades. Several respective cloud platforms are already in our sights, e.g., Microsoft Azure ML [9], Amazon AWS ML [15] and Google Cloud AI [8]. Nevertheless, these platforms only apply intuitive schedulers, for example, FIFO or Dominant Resource Fairness Scheduling (DRF) [10], which can not maximize resource utilization. Other recent proposed schedulers [24], [25], [26], [27], [28] focus on maximizing resource utilization, but usually do not take the data transmission consumption into consideration. However, in the edge network, this type of consumption can not be overlooked due to the bandwidth is scarce and expensive. Our previous work [29] identifies UDL jobs' distinct characteristics (training data is geo-distributed over sites), and discusses the dynamic deployment of workers and PSs for UDL jobs to minimize the overall training time. Nonetheless, in this paper, we investigate a totally different problem focusing on overall utility maximization (social welfare). Besides, to avoid overpricing or underpricing, tailored pricing functions based on market rules are further presented.

Auction Mechanism in Edge/Cloud: Wu et al. [16] propose a specific auction to maximize social welfare for a mobile

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

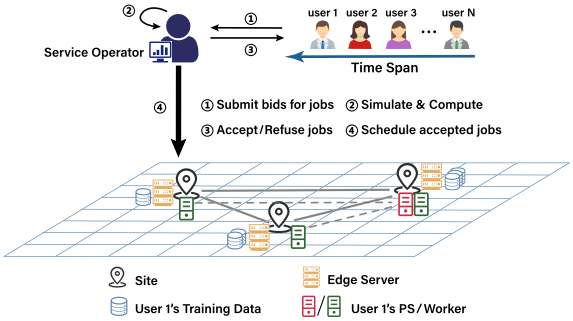

Fig. 1. System of unbiased distributed learning.

edge computing (MEC) system. Mashhadi et al. [18] design a truthful auction for MEC edge servers that can maximize overall profit without compromising mobile devices' energy consumption and delay requirements. Lin et al. [17] present an auction framework to effectively allocate resources to maximize edge servers' utilities meanwhile ensuring the latency constraint for computing tasks of end-devices. In [19], an online auction is developed to maximize cloud provider's revenue in a cloud computing system. Zhang et al. [20] first discuss the heterogeneity of user demands within an auction scheme. Li et al. [21] focus on resource pricing and design an auction to achieve trade-off the utilities between users and cloud service provider. Zhang et al. [30] investigate the time-varying cloud resource allocation problem. Overall, the above work and other non-mentioned auction mechanisms [31], [32], [33] cannot be applied to price UDL jobs as the job training time is uncertain and highly depends on the job placement.

III. System Model

A. System Overview

Auction Overview: Fig. 1 depicts an auction scenario which involves two types of entities: the service operator (auctioneer) and the users (bidders). The operator owns the edge network and the training data, and receives bids from users to train *unbiased distributed learning* (UDL)[2] jobs in its edge system. Once it received the bids submitted by users, the operator conducts a schedule policy for jobs, and calculates the required payment. Then, the operator allocates resources for the accepted job, and charges the user accordingly. Assume that I jobs arrive and request for training within T time slots. Note that $\mathcal{X}$ indicates the integer set $\{1, 2, \ldots, X\}$. Table I lists some important notations.

Service Operator: The service operator serves as the owner and manager of resources in the edge distributed learning system. The system involves R geo-distributed "sites", e.g., an edge cloudlet or edge server [29]. Each site collects and provides training data for UDL jobs, which are targeted at different smart applications, such as, intelligent video surveillance and smart city. In addition, each site r can offer C_r^k units of type-k resource and connects with other sites by edge networks. Let K denote the number of resource types, e.g., GPU, CPU and disk storage. Since radio resources in the edge network are scarce and expensive, as a result, the operator imposes a limit of the total data rate for transmission, denoted by B.

[2]Different from traditional modes that DL job associates with its own input dataset, UDL jobs' datasets are collected from edge sites, to support the unbiasedness of DL model [3].

TABLE I
List of Notations

I	# of jobs	T	# of time slots
a_i	job i's arrival time	$\hat{t}_i$	completion time of job i
x_i	accept job i or not	$f_i(\cdot)$	job i's utility function
P_i	processing capacity of job i's worker		
D_i^r	the size of the training data for job i in site r		
B	overall volume (data rate) of the amount of data transmission per time slot		
$z_i^r(t)$	whether job's PS is placed in site r at t		
$g_i^r(t)$	# of job i's workers in site r in t		
C_r^k	the capacity of type-k resource in site r		
λ_i	the size (MB) of the parameter exchanged between a worker and the PS of job i per time slot		
$h_i^{rr'}(t)$	the size of the training data of job i transmitted from site r to site r' at t		
θ	the size (MB) of one data chunk		
$w_i^k\ (s_i^k)$	the amount of type-k resource occupied by one worker (PS) of job i		
x_{il}	select schedule l for job i or not		
$\psi_{il}(t)$	the amount of data transmitted between site r and r' at t in job i's schedule l		
$\varphi_{il}^{r,k}(t)$	the amount of type-k resource in site r currently occupied by job i according to its schedule l		
$\rho(t)$	the size of data transmitted at time t		
$\eta_r^k(t)$	the amount of allocated type-k resource of site r at t		
$\alpha(t)$	unit price for data transmission (MB) at time t		
$\beta_r^k(t)$	unit price of the type-k resource of site r at t		
$M_i^r(t)$	remaining size of training data of job i in site r at t		
$\gamma_d(\gamma_k)$	the base of the data transmission (resources) price functions		

When a new job arrives, the service provider has to make multiple decisions, such as: i) whether accept this job; ii) where the PS of this job should be placed; iii) the allocated number of workers for this job; iv) how to transmit the training data among edge sites for this job.

Users: Users submit their bids for training jobs once they arrive. For ease of analysis, suppose one user submit one job at a time. Thus, we do not distinguish a user and a job below. User i submits bid information, which is a tuple Π_i

$$\Pi_i = \{a_i, f_i(\cdot), \{D_i^r\}_{\forall r}, E_i, \{w_i^k, s_i^k\}_{\forall k}\}, \tag{1}$$

where a_i indicates the arrival time of job i, $f(\cdot)$ denotes the specific utility function of job i, which is non-negative and non-increasing with its execution duration $(\hat{t}_i - a_i)$ [34], [35]. Generally, one UDL job's utility is customized by its user according to the latency-sensitive requirement (i.e., completion time) and the characteristics of its workload. To clarity, assuming that data chunk is set to be the unit of data. For job i, D_i^r represents the required data size (the number of data chunks) in site r. For good performance, all data needs to be trained for E_i epochs. $w_i^k\ (s_i^k)$ represents the amount of type-k resource occupied by one worker (PS) of job $i, \forall k \in \mathcal{K}$. Let P_i indicate the processing capacity of job i's worker, i.e., the number of data chunks job i's worker can process in one time slot.

Parameter Server Architecture: In order to reduce bandwidth consumption, we adopt the *data parallel* training and the PS architecture [36], rather than requiring all training data across

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

geo-distributed sites to train centrally [37]. Workers and PSs are configured as virtual machines (VMs) or containers. Assuming that each UDL job is allocated with one PS, which in practice can represent various PS instances within one site. We use *synchronous training* to guarantee the model convergence [38] due to the fact that *synchronous training* can obtain a model with comparatively higher level of accuracy compared to asynchronous training [39]. Typically, one time slot will be longer than a training epoch in length, so one can perform training at each time slot. One time slot, for instance, will be for 30 minutes or more. To meet the demand for training data, at beginning of each time slot, one needs to redistribute training data (workload) among sites for processing at the current time slot. For adopting Mini-batch Gradient Descent [40], each data chunk will be split up into some equal-sized mini-batches. Iteratively, workers calculate and transmit model gradients (directions of changes) to the PS. The PS updates the model after received all workers' model parameters, and then sends back to all workers. Once all mini-batches have been processed for the predefined number of epochs, workers will further compute gradients using the subsequent mini-batch. Let λ_i denote the total size (MB) of the parameter transmitted between a worker and the PS per slot.

Auction Preliminary: Here, to make everyone happy in the long run, we focus on the whole system's overall utilities, that is, aiming to achieve *social welfare* maximization. To achieve this, eliciting truthful information from users is extremely necessary. In practice, due to the egocentric characteristic of users, they will attempt to maximize their own payoffs. More specifically, to obtain a higher payoff, users may try to lie about the real valuation (function) of their job bids in terms of completion time, denoted as $v_i(\cdot)$. Here, we will introduce the following definitions for auction design in terms of truthfulness and social welfare.

Definition 1(Truthful Auction): An auction is truthful if and only if every user's payoff is maximized by reporting its true valuation, i.e., for all $f_i(\cdot) \neq v_i(\cdot)$, $u_i(v_i(\cdot)) \geq u_i(f_i(\cdot))$.

Definition 2(Social Welfare): Note that the payoff of job i is $u_i(f_i(\cdot)) = v_i(\cdot) - p_i$ if its bid is accepted, and 0 otherwise. The operator's revenue is the total payment of all accepted jobs, i.e., $\sum_{i\in\mathcal{I}} p_i x_i$. Here, the social welfare of the auction is defined as the aggregate utilities of the operator and users, and equals $\sum_{i\in\mathcal{I}}(v_i(\cdot) - p_i)x_i + \sum_{i\in\mathcal{I}} p_i x_i$, i.e., $\sum_{i\in\mathcal{I}} v_i(\cdot)x_i$.

B. Problem Formulation

To clarify, we define the decisions made by the operator as: i) x_i, a binary variable which means whether job i's bid is accepted ($x_i = 1$) or not ($x_i = 0$); ii) $z_i^r(t)$, a binary variable which indicates whether job i's PS is placed in site r at time slot t or not; iii) $g_i^r(t) \in \mathbb{N}^+$, the allocated number of workers for job i in site r at $t, \forall i \in \mathcal{I}, \forall r \in \mathcal{R}$; iv) $h_i^{rr'}(t) \in \mathbb{N}^+$, the number of data chunks for job i transmitted from site r to r' at time t; v) $p_i \geq 0$, the required payment for user i once the operator accepts his bid ($x_i = 1$).

The social welfare maximization problem can be expressed as follows under truthful bidding ($v_i(\cdot) = f_i(\cdot)$).[3]

$$\text{maximize} \quad \sum_{i\in\mathcal{I}} x_i f_i(\hat{t}_i - a_i), \tag{2}$$

subject to

$$\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\sum_{r'\in\mathcal{R}} h_i^{rr'}(t) = \sum_{r\in\mathcal{R}} D_i^r x_i, \quad \forall i \in \mathcal{I}, \tag{2a}$$

$$P_i g_i^{r'}(t) \geq E_i \sum_{r\in\mathcal{R}} h_i^{rr'}(t), \quad \forall i \in \mathcal{I}, \forall r' \in \mathcal{R}, \forall t \in \mathcal{T}, \tag{2b}$$

$$\sum_{r\in\mathcal{R}} z_i^r(t) = x_i, \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \tag{2c}$$

$$\sum_{i\in\mathcal{I}}\sum_{r,r'\in\mathcal{R}} (\theta h_i^{rr'}(t) + \lambda_i g_i^r(t)\mathbb{I}(z_i^r(t) = 0)) \leq B, \forall t \in \mathcal{T}, \tag{2d}$$

$$\sum_{i\in\mathcal{I}} (w_i^k g_i^r(t) + s_i^k z_i^r(t)) \leq C_r^k, \quad \forall k \in \mathcal{K}, \forall r \in \mathcal{R}, \forall t \in \mathcal{T}, \tag{2e}$$

$$\hat{t}_i = \arg\max_{t\in\mathcal{T}} \left(\sum_{r\in\mathcal{R}} g_i^r(t) > 0 \right), \quad \forall i \in \mathcal{I}, \tag{2f}$$

$$g_i^r(t) = h_i^{rr'}(t) = z_i^r(t) = 0, \quad \forall t : t < a_i, \tag{2g}$$

$$g_i^r(t), h_i^{rr'}(t) \in \{0, 1, 2, \dots\}, \quad \forall i \in \mathcal{I}, \forall r, r' \in \mathcal{R}, \forall t \in \mathcal{T}, \tag{2h}$$

$$x_i, z_i^r(t) \in \{0, 1\}, \quad \forall i \in \mathcal{I}, \forall r \in \mathcal{R}, \forall t \in \mathcal{T}. \tag{2i}$$

Constraint (2a) ensures that the amount of transmitted data equals the amount of job i's training data. Specially, $h_i^{rr}(t)$ denotes the *local training data* in site r at t, which means it does not need to be transmitted at t. Constraint (2b) guarantees that job i's workers deployed in site r' at t is sufficient for handling E_i epochs of data in site r'. Constraint (2c) indicates that each job has one PS if accepted. Constraint (2d) illustrates that the amount of training data transmitted and the amount of parameter transmitted between sites limited by the data rate B. Given the edge network's inherent dynamic conditions like interference and fading, our constraint (2d) focuses on data volume, providing a more stable metric than uncontrollable traditional bandwidth. Utilizing data volume allows us to indirectly constrain on associated physical resources, such as bandwidth or transmission power, ensuring that the system operates within feasible and realistic limits. Here, we consider that there is no

[3]Specially, decision variable p_i does not appear in problem (2) because of the definition of social welfare (Definition 2).

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

need to transmit parameters if the PS and workers are deployed in the same site. We specify this case by using $\mathbb{I}(z_i^r(t)=0)$, where $\mathbb{I}(\cdot)$ is an indicator function. The resource constraint of sites for deploying workers and PS is demonstrated by (2e). The completion time of one job is defined in constraint (2f), that is, defining the completion time of a job as the largest time slot that this job still need workers to train.

Challenges: Problem (2) is a mixed integer nonlinear programming (MINLP), which is NP-hard [41], [42]. This statement can be easily supported by the following reformulated problem (3), which can be transformed from a classical Knapsack problem in a polynomial time. Furthermore, the problem (2) involves integer variables which are dependent on each other, and a non-conventional constraint (2f). In this regard, there is a need to make online decisions for deploying workers and PSs, and transmitting data, without any future information.

IV. Algorithm Design

In this section, we develop an auction-based scheduling framework, *Eris*, to schedule UDL jobs. *Eris* aims to achieve social welfare maximization. The following is the main idea.

i. In Section IV-A, we reformulate problem (2) into an integer linear program (ILP) by introducing exponential number of variables. Then, we construct its dual problem to handle the primal variables. By applying the theory of complementary slackness, we convert our goal to compute each job's payoff, which means jobs with positive payoff will be accepted, otherwise rejected. The computation of jobs' payoff is based on resource pricing and its corresponding assigned resources.
ii. In Section IV-B, we analyze the traditional pricing mechanisms in the market. Then we design various tailored price functions for resources based on market rules.
iii. In Section IV-C, we discuss how to deploy workers and the PS for one job to reduce resource consumption. *Eris* further decomposes problem (7) to a series of one-shot problems (8). A dynamic programming (DP) algorithm is applied to divide the total training data into time slots within the range $[a_i, \hat{t}_i]$. For each one-shot problem, *Eris* conducts data transmission according to a tailored metric. Then *Eris* deploys workers in a greedy fashion according to the resource cost of one worker. Finally, the PS is placed to the site which has the maximum number of workers to reduce the communication cost.

A. Problem Reformulation and Auction Framework

Recall that searching for a feasible schedule for each job is the ultimate goal of the service operator. Therefore, to address problem (2), we consider reformulating it into the following 0-1 integer linear program (ILP) by encoding three types of integer variables $z_i^r(t)$, $g_i^r(t)$ and $h_i^{rr'}(t)$ into one schedule variable l.

$$\text{maximize} \quad \sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}_i} x_{il} f_i(\hat{t}_{il}-a_i), \tag{3}$$

subject to

$$\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}_i}\psi_{il}(t)x_{il}\le B, \quad \forall t\in\mathcal{T}, \tag{3a}$$

$$\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}_i}\varphi_{il}^{r,k}(t)x_{il}\le C_r^k, \quad \forall k\in\mathcal{K},\forall r\in\mathcal{R},\forall t\in\mathcal{T}, \tag{3b}$$

$$\sum_{l\in\mathcal{L}_i} x_{il}\le 1, \quad \forall i\in\mathcal{I}, \tag{3c}$$

$$x_{il}\in\{0,1\}, \quad \forall l\in\mathcal{L}_i,\forall i\in\mathcal{I}. \tag{3d}$$

In ILP (3), schedule l represents one deployment result for job i, which consists of integer variables $z_i^r(t)$, $g_i^r(t)$ and $h_i^{rr'}(t)$. $\mathcal{L}_i$ denotes the set containing job i's all feasible schedules that satisfies constraint (2a)–(2c) and (2f). For consistency, binary variable x_i is reshaped to x_{il}, which denotes whether select job i's schedule l or not. And $\hat{t}_{il}$ represents the completion time of job i according to schedule l. Likewise, $\psi_{il}(t)$ represents the amount of data transmitted between site r and r' at t in job i's schedule l, i.e., $\psi_{il}(t)=\sum_{r,r'\in\mathcal{R}}\theta h_{il}^{rr'}(t)+\lambda_i g_{il}^r(t)\mathbb{I}(z_{il}^r(t)=0)$. $\varphi_{il}^{r,k}(t)$ indicates the amount of type-k resource in site r currently occupied by job i according to its schedule l, i.e., $\varphi_{il}^{r,k}(t)=w_i^k g_{il}^r(t)+s_i^k z_{il}^r(t)$. In addition, variables $z_{il}^r(t)$, $g_{il}^r(t)$ and $h_{il}^{rr'}(t)$ represent the corresponding value within schedule l, respectively. Constraint (3a) and (3b) are equivalent to constraint (2d) and (2e), respectively. Then constraint (3c) and (3d) are corresponding to constraints (2a)–(2c). Thus, ILP (3) is equivalent to problem (2).

Dual Problem: Obviously, the number of feasible schedules of each job is potentially exponential because of combinatorial nature of these variables. To tackle the exponential scale of introduced variables, we construct its dual problem (4) by relaxing the binary variable x_{il} to $x_{il}\ge 0$. Dual variables $\alpha(t)$, $\beta_r^k(t)$ and μ_i are associated with constraint (3a), (3b) and (3c), respectively.

$$\text{minimize} \quad \sum_{i\in\mathcal{I}}\mu_i+\sum_{t\in\mathcal{T}}B\alpha(t)+\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}C_r^k\beta_r^k(t), \tag{4}$$

subject to

$$\mu_i\ge f_i(\hat{t}_{il}-a_i)-\sum_{t\in\mathcal{T}}\psi_{il}(t)\alpha(t)-\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}\varphi_{il}^{r,k}(t)\beta_r^k(t), \quad \forall i\in\mathcal{I},\forall l\in\mathcal{L}_i, \tag{4a}$$

$$\mu_i,\alpha(t),\beta_r^k(t)\ge 0, \quad \forall i\in\mathcal{I},\forall r\in\mathcal{R},\forall t\in\mathcal{T}. \tag{4b}$$

Design Rationale: By interpreting the dual variable $\alpha(t)$ as the unit price for data transmission (MB) at t, $\sum_{t\in\mathcal{T}}\psi_{il}(t)\alpha(t)$ indicates the data transmission cost of job i. Similarly, $\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}\varphi_{il}^{r,k}(t)\beta_r^k(t)$ represents the total cost of resources occupied by job i with schedule l, when $\beta_r^k(t)$ recognizes as the unit price of the type-k resource of site r at time t. So the right hand side (RHS) of (4a) means job i's utility minus the overall cost. In this regard, μ_i represents the received payoff of job i if it is accepted. To minimize the objective of the dual problem, we set the dual variable μ_i to be the maximum value between 0 and the RHS of (4a) based on the optimal

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Auction Framework *Eris*.

Input: $T, C_r^k, B, \forall k \in \mathcal{K}, \forall r \in \mathcal{R}$
Output:
$x_i, g_i^r(t), z_i^r(t), h_i^{rr'}(t), \forall i \in \mathcal{I}, \forall r, r' \in \mathcal{R}, \forall t \in \mathcal{T}$
1: Initialize
$g_i^r(t) = 0, z_i^r(t) = 0, h_i^{rr'}(t) = 0, \rho(t) = 0, \eta_r^k(t) = 0, \alpha(t) = \alpha(0), \beta_r^k(t) = \beta_r^k(0), \quad \forall i, \forall r, r', \forall k, \forall t$
2: **for** $i = 1, 2, \ldots, I$ **do** /*new jobs*/
3: Set$(l^*, \mu_i) =$
$SA(a_i, E_i, \{D_i^r\}_{\forall r}, \{\rho(t)\}_{\forall t}, \{\eta_r^k(t)\}_{\forall r,k,t})$
4: **if** $\mu_i > 0$ **then**
5: Update $\rho(t) = \rho(t) + \psi_i(t), \eta_r^k(t) = \eta_r^k(t) + \varphi_i^{r,k}(t), \alpha(t) = \alpha(\rho(t)), \beta_r^k(t) = \beta_r^k(\eta_r^k(t)), \forall r, r', \forall k, \forall t$
6: Set $x_i = 1$ and deploy job i according to schedule l^*
7: **else**
8: Set $x_i = 0$ and reject job i
9: **end if**
10: **end for**

schedule l^*.

$$\mu_i = \max\{0, \max_{l \in L_i} \text{RHS of (4a)}\}. \tag{5}$$

If $\mu_i > 0$, the operator accepts and allocates resources for job i according to its schedule l^* ($x_{il^*} = 1$); otherwise, the operator rejects ($x_{il} = 0, \forall l \in \mathcal{L}_i$). The explanation is that jobs' value can cover the overall cost, which ensures the jobs' owners are willing to train their jobs. Besides, the operator attempts to accept those jobs with larger value and less resource consumption.

Auction Framework: The main workflow of *Eris* is presented in Algorithm 1. The subroutine Algorithm 2 computes the optimal schedule l^* for each newly arrived job (Line 3). If the value of μ_i is larger than 0, the operator accepts the job ($x_i = 1$) and updates the current price of resources or data by using well-designed price functions (lines 4–7), discussed in Section IV-B.

B. Price Function Design

Before finding the optimal schedule for jobs, let us first discuss the marginal price of data and resources, i.e., variables $\alpha(t)$ and $\beta_r^k(t)$. In market economics, resource price always increases with the amount of resources occupied. To be specific, when there are enough resources available, resource prices should be decreased to encourage consumption. When demand for a resource exceeds supply, service providers will inevitably raise the price in order to relieve the strain on supply. In this paper, we adopt an exponential function as the basic expression to depict this market rule. Here, let $\rho(t)$ denote the amount of data transmitted at time t, i.e., $\sum_{i \in \mathcal{I}_t} \sum_{r,r' \in \mathcal{R}} \theta h_i^{rr'}(t) + \lambda_i g_i^r(t) \mathbb{I}(z_i^r(t) = 0)$. And $\mathcal{I}_t$ indicates a set of accepted jobs before job i arrived. Additionally, $\eta_r^k(t)$ represents the amount of allocated type-k resource of site r at time t, i.e., $\sum_{i \in \mathcal{I}_t} \sum_{r \in \mathcal{R}} w_i^k g_i^r(t) + s_i^k z_i^r(t)$. Therefore, the price functions of data transmission/resources can be formulated as

$$\alpha(\rho(t)) = (\gamma_d)^{\frac{\rho(t)}{B}} - 1, \quad \beta_r^k(\eta_r^k(t)) = (\gamma_k)^{\frac{\eta_r^k(t)}{C_r^k}} - 1. \tag{6}$$

Here, let us discuss several characteristics of our tailored price functions. If there are no resources used (i.e., $\rho(t) = 0$) or no transmitted data (i.e., $\eta_r^k(t) = 0$), then $\alpha(t) = \beta_r^k(t) = 0, \forall r, \forall k, \forall t$. In this case, all jobs would be accepted, to incentive more people to submit their jobs in this auction, and increase the possibility of accepting jobs with larger utility for the operator. If there are no available resources or no data budget left, $\alpha(t) = \gamma_d - 1, \beta_r^k(t) = \gamma_k - 1, \forall r, \forall k, \forall t$. This means that these two values should be large enough to avoid any jobs from being accepted by the operator, i.e., $f_i(\hat{t}_{il} - a_i) - \sum_{t \in \mathcal{T}} (\gamma_d - 1)\psi_{il}(t) - \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}} \sum_{k \in \mathcal{K}} (\gamma_k - 1)\varphi_{il}^{r,k}(t) < 0$. A feasible solution is to set $\frac{f_i(\hat{t}_{il} - a_i)}{\sum_{t \in \mathcal{T}} \psi_{il}(t)} \leq \gamma_d - 1$, $\frac{f_i(\hat{t}_{il} - a_i)}{\sum_{r \in \mathcal{R}} \sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{T}} \varphi_{il}^{r,k}(t)} \leq \gamma_k - 1, \forall i$ by assuming that $\gamma_k - 1 \geq 1, \gamma_d - 1 \geq 1$, i.e., $\frac{f_i(\hat{t}_{il} - a_i)}{\sum_{t \in \mathcal{T}} \psi_{il}(t)} \geq 1$, $\frac{f_i(\hat{t}_{il} - a_i)}{\sum_{r \in \mathcal{R}} \sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{T}} \varphi_{il}^{r,k}(t)} \geq 1, \forall i$. It is reasonable because there are no sufficient resources to handle more jobs. Moreover, such assumptions can prevent users from submitting jobs and enable the operator to escape peak load quickly.

C. Sub-Problem of Finding Schedule l

The sub-problem of finding an optimal schedule l^* for each job i can be expressed as follows.

$$\max_{\hat{t}_i, \mathbf{g}, \mathbf{z}, \mathbf{h}} f_i(\hat{t}_i - a_i) - \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}} \sum_{k \in \mathcal{K}} (w_i^k g_i^r(t) + s_i^k z_i^r(t)) \beta_r^k(t) - \sum_{t \in \mathcal{T}} \sum_{r,r' \in \mathcal{R}} (\theta h_i^{rr'}(t) + \lambda_i g_i^r(t) \mathbb{I}(z_i^r(t) = 0)) \alpha(t), \tag{7}$$

subject to

$$\psi_i(t) \leq B - \rho(t), \forall t \in \mathcal{T}, \tag{7a}$$

$$\varphi_i^{r,k}(t) \leq C_r^k - \eta_r^k(t), \forall r \in \mathcal{R}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}, \quad (2a) - (2c), (2f) - (2i), \text{where } x_i = 1. \tag{7b}$$

Constraint (7a) and constraint (7b) are equivalent to constraint (3a) and (3b), respectively. To solve problem (7), we consider fixing and enumerating the completion time $\hat{t}_i$ to simplify problem (7). Then we further apply DP to divide the total workload D_i (i.e., $\sum_r \mathcal{D}_i^r$) into $D_i(t), \forall t \in [a_i, \hat{t}_i]$. Consequently, problem (7) can be simplified as a series of following one-shot problems.

$$\min cost(t, D_i(t)) = \sum_{r,r' \in \mathcal{R}} (\theta h_i^{rr'}(t) + \lambda_i g_i^r(t) \mathbb{I}(z_i^r(t) = 0)) \alpha(t) + \sum_{r \in \mathcal{R}} \sum_{k \in \mathcal{K}} (w_i^k g_i^r(t) + s_i^k z_i^r(t)) \beta_r^k(t), \tag{8}$$

subject to

$$(7a) - (7b), (2b) - (2c), (2g) - (2i), \text{where } x_i = 1.$$

Algorithm Details: The details of simplifying and computing optimal schedule l^* is given in Algorithm 2. In lines 2–7,

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2: Scheduling Algorithm *SA*.

Input: $a_i, E_i, D_i^r, \rho(t), \eta_r^k(t), \forall k \in \mathcal{K}, \forall r \in \mathcal{R}, \forall t \in \mathcal{T}$
Output: l^*, μ_i
1: Initialize $l^* = \varnothing$, $\mu_i = 0$
2: **for** $\hat{t}_i = a_i$ to T **do**
3: $(cost^*, l) = DPC(\hat{t}_i, \sum_{r\in\mathcal{R}} \mathcal{D}_i^r)$
4: **if** $\mu_i < f_i(\hat{t}_i - a_i) - cost^*$ **then**
5: $\mu_i = f_i(\hat{t}_i - a_i) - cost^*, l^* = l$
6: **end if**
7: **end for**
8: Return l^*, μ_i
9: **function** $DPCT_i, D_i$
10: $cost_i = \infty, l = \varnothing$
11: **for** $d = 0$ to D_i **do**
12: $(cost_t, \{g_i^r(T_i)\}_{\forall r}, \{z_i^r(T_i)\}_{\forall r}, \{h_i^{rr'}(T_i)\}_{\forall r,r'}) = DPC\_T(T_i, d)$
13: $(cost, l') = \text{DPC}(T_i - 1, D_i - d)$
14: **if** $cost_i > cost_t + cost$ **then**
15: $cost_i = cost_t + cost, l = l' \cup \{\{g_i^r(T_i)\}_{\forall r}, \{z_i^r(T_i)\}_{\forall r}, \{h_i^{rr'}(T_i)\}_{\forall r,r'}\}$
16: **end if**
17: **end for**
18: Return $cost, l$
19: **end Function**

for new arrived job i, Algorithm 2 enumerates its $\hat{t}_i$ within the range $[a_i, T]$. In line 5, Algorithm 2 obtains the optimal schedule for each job with the fixed $\hat{t}_i$ by function *DPC*. Through comparing payoff μ_i at different $\hat{t}_i$ achieved by *DPC*, the optimal schedule l^* can be returned (lines 4–6). Lines 9–18 achieve a DP function: $DPC(\hat{t}_i, D_i) = \min_{d\in[0,D_i]} DPC\_T(\hat{t}_i, d) + DPC(\hat{t}_i - 1, D_i - d)$. The workload to be finished within time slot $\hat{t}_i$ will be enumerated from 0 to D_i to narrow the scheduling scale (lines 11–12). The remaining workload $D_i - d$ will be handled within $[a_i, \hat{t}_i - 1]$ (line 13). Lines 14–16 attempt to identify the corresponding schedule with the smallest cost. Similarly, obtaining the corresponding optimal schedule at a narrower time window $[a_i, \hat{t}_i - 1]$ can be addressed. For time slot t within $[a_i, \hat{t}_i - 1]$, we will enumerate the workload and further narrow the scheduling scale recursively until reach the arriving time a_i. The deployment algorithm is discussed in Section IV-D.

D. Deployment Algorithm

The key to solve problem (8) is to decide where to deploy workers and the PS, and to determine data transmission to minimize the overall cost. This can be addressed optimally by a simple greedy algorithm, which is discussed in Algorithm 3. And its optimality will be shown in Theorem 4.

Here, we consider redistributing the training data across sites to accommodate workers and the PS's deployment. Algorithm 3 makes decisions for deploying workers and PSs at t to fulfill workload $D_i(t)$, and calculates the required data transmission. The data redistribution is based on a tailored metric, which is defined as below.

$$\Omega_i^{rr'}(t) = \theta\alpha(t) - \frac{E_i}{P_i}\sum_{k\in\mathcal{K}} w_i^k(\beta_r^k(t) - \beta_{r'}^k(t)), \tag{9}$$

where the first term denotes the cost of transmitting one data chunk, and the second term represents the reduced resource cost of one data chunk if we transmit it from site r to r'. The rationale is that we aim to transmit data to sites with lower resource cost to train if the reduced resource cost can cover the cost introduced by data transmission. Note that we will stop transmitting data if the value of $\Omega_i^{rr'}(t)$ is smaller than 0, i.e., the cost of transmitting one data chunk is larger than the corresponding reduced resource cost. Therefore, we first sort pairs of sites (i.e., $\{r, r'\}$) in non-decreasing order according to the metric $\Omega_i^{rr'}(t)$ in line 2. Then we enumerate each pair to determine whether the operator needs to transmit data between sites or not (lines 2–11). In particular, line 5 calculates $h_i^{r_j r'_j}(t)$ by using the following equation.

$$\begin{aligned} h_i^{r_j r'_j}(t) = \min\{M_i^{r_j}(t) - \sum_{r'\in\mathcal{R}} h_i^{r_j r'}(t), \left\lfloor \frac{B - \rho(t)}{\theta} \right\rfloor, \\ D_i(t) - \sum_{j'=1}^{j-1}(M_i^{r'_{j'}}(t) + \sum_{r\in\mathcal{R}} h_i^{r r'_{j'}}(t)), \\ \frac{P_i}{E_i}\min_{k\in\mathcal{K}} \left\lfloor \frac{C_{r'_j}^k - \eta_{r'_j}^k(t)}{w_i^k} \right\rfloor - M_i^{r'_j}(t) - \sum_{r\in\mathcal{R}} h_i^{r r'_j}(t)\}, \end{aligned} \tag{10}$$

where the first term represents the remaining amount of job i's data in site r_j. The second term guarantees constraint (7a). The third term indicates the remaining workload needed to be processed currently. In addition, the last term represents that the number of data chunks that site r'_j can still handle currently except its current training data volumes. In lines 6–8, we decide whether transmitting data from site r_j to r'_j is feasible through the value of $h_i^{r_j r'_j}(t)$. We reset $h_i^{r_j r'_j}(t)$ to 0 and exclude all pairs includes site r'_j if transmitting data is unnecessary, i.e., $h_i^{r_j r'_j}(t) < 0$.

After finishing the data re-distribution, we consider deploying workers via a greedy fashion according to the unit price of one worker of job i, i.e., $\sum_{k\in\mathcal{K}} w_i^k \beta_r^k(t)$. Lines 13–15 compute the corresponding number of workers on each site. Here, we reserve resources for the PS in each site to reduce the complexity. This is reasonable because the resources occupied by workers and PS are essentially different. Workers need GPU to train [11], however PSs aggregate parameters using CPU. So reserving resources for the PS would not affect the deployment of workers. To reduce the cost of parameter exchanged, we greedily deploy the PS on the site, which has the maximum number of workers (line 16). If not enough workers or violating constraint (7a), fulfilling workload $D_i(t)$ at time t is infeasible (lines 17–19); otherwise, we return the overall cost and corresponding schedule.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

Algorithm 3: Deployment Algorithm *DPC_T*.

Input: $t, D_i(t)$;
Output: $cost_t, g_i^r(t), z_i^r(t), h_i^{rr'}(t), \forall r, r' \in \mathcal{R}$
1: Initialize $g_i^r(t) = 0, z_i^r(t) = 0, h_i^{rr'}(t) = 0, \forall r, r' \in \mathcal{R}$
2: Sort all pairs of sites according to $\Omega_i^{rr'}(t)$ in non-decreasing order into $\{r, r'\}_1, \{r, r'\}_2, \ldots, \{r, r'\}_{\frac{R(R-1)}{2}}$
3: **for** $j = 1$ to $\frac{R(R-1)}{2}$ **do**
4: **if** $\Omega_i^{rr'}(t) < 0$ **then** /* whether transmit data or not */
5: Compute $h_i^{r_j r'_j}(t)$ using (10)
6: **if** $h_i^{r_j r'_j}(t) < 0$ **then** /* exceeding resource limits */
7: $h_i^{r_j r'_j}(t) = 0$
8: Exclude all pairs $\{r, r'\}_j$ that include site r'_j
9: **end if**
10: **end if**
11: **end for**
12: Sort sites in $\mathcal{R}$ according to $\sum_{k\in\mathcal{K}} w_i^k \beta_r^k(t)$ in non-decreasing order into $r_1, r_2, \ldots, r_R$
13: **for** site $j = 1$ to R **do** /* deploy workers */
14: $g_i^{r_j}(t) = \min\{\lceil \frac{E_i}{P_i}(M_i^{r_j}(t) + \sum_{r\in\mathcal{R}} h_i^{rr_j}(t))\rceil, \min_{k\in\mathcal{K}}\lfloor \frac{C_{r_j}^k - \eta_{r_j}^k(t)}{w_i^k}\rfloor, \lceil \frac{E_i D_i(t)}{P_i}\rceil - \sum_{r\in\mathcal{R}} g_i^r(t)\}$
15: **end for**
16: $\hat{r} = \arg\max_{r\in\mathcal{R}}\{g_i^r(t)\}$, $z_i^{\hat{r}}(t) = 1$
17: **if** $\sum_{r\in\mathcal{R}} g_i^r(t) < \lceil \frac{E_i D_i(t)}{P_i}\rceil$ or $B - \rho(t) - \theta\sum_{r,r'\in\mathcal{R}} h_i^{rr'}(t) - \lambda_i \sum_{r:r\neq\hat{r}} g_i^r(t) < 0$ **then**
18: Return $cost_t = +\infty$, $\{g_i^r(t)\}_{\forall r}$, $\{z_i^r(t)\}_{\forall r}$, $\{h_i^{rr'}(t)\}_{\forall r,r'}$
19: **end if**
20: $cost_t = \sum_{r,r'\in\mathcal{R}} \theta h_i^{rr'}(t)\alpha(t) + \sum_{r:r\neq\hat{r}} \lambda_i g_i^r(t)\alpha(t) + \sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}(w_i^k g_i^r(t) + s_i^k z_i^r(t))\beta_r^k(t)$
21: Return $cost_t$, $\{g_i^r(t)\}_{\forall r}$, $\{z_i^r(t)\}_{\forall r}$, $\{h_i^{rr'}(t)\}_{\forall r,r'}$

V. Theoretical Analysis

In Section V, we verify several key properties of *Eris* including the correctness, individual rationality, truthfulness, competitive ratio and time complexity.

Theorem 1: Our proposed auction *Eris* produces a feasible solution to the original sub-problem (2).

Proof: We first show that a feasible solution of sub-problem (8) can be obtained by using Algorithm 3. For each pair, we calculate its corresponding transmitted data by using (10), which can satisfy constraint (7a), (2d), and (2h). Line 14 of Algorithm 3 computes $g_i^r(t)$ according to constraint (7a), (7b), (2b), and (2e). And the placement of the PS also satisfies constraint (2c). So Algorithm 3 produces a feasible solution of sub-problem (8). Function *DPC* can guarantee constraint (2a) will not be violated. In lines 2–7 of Algorithm 2, the *for* loop enumerates $\hat{t}_i$, which can satisfy constraint (2f). Therefore, we complete the proof. □

Theorem 2(Individual rationality): *Eris* can achieve individual rationality.

Proof: To prove this, we need to ensure that the utilities of users and the profit of the service provider are both non-negative. For users, this requirement can be easily satisfied because (5) ensures that if one user' utility $\mu_i > 0$, the operator accepts and allocates resources for job i according to its schedule l^* ($x_{il^*} = 1$); otherwise, the operator rejects ($x_{il} = 0, \forall l \in \mathcal{L}_i$). For the service provider, its profit equals to the total payment of all accepted jobs, i.e., $\sum_{i\in I} p_i x_i$, which always be positive. Therefore, we finish this proof. □

Theorem 3(Truthfulness): *Eris* is a truthful auction.

Proof: This theorem can be verified through analyzing the truthfulness of all elements in the bid tuple.

Truthfulness in the arrival time a_i: To start with, we discuss that user i has no motivation to report an earlier arrival time of the job. In fact, user i's job cannot arrive at the reported/eariler time. In this case, the service operator can directly discard this job when this happened. Besides, additional punishment mechanisms can be applied to avoid this. If user i bids with a later arrival time, it does not actually affect the service operator to schedule jobs as expected. Intuitively, users also have no motivation to do so because it would prolong the completion time of jobs. Therefore, we finish this part of proof.

Truthfulness in utility functions $f_i(\cdot)$: Recall that *Eris* is based on the posted pricing mechanisms [43]. For each accepted job i, its payment is computed as $p_i = \sum_{t\in\mathcal{T}}\sum_{r,r'\in\mathcal{R}} \theta h_i^{rr'}(t)\alpha(t) + \sum_{t\in\mathcal{T}}\sum_{r:r\neq\hat{r}} \lambda_i g_i^r(t)\alpha(t) + \sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}(w_i^k g_i^r(t) + s_i^k z_i^r(t))\beta_r^k(t)$. We can see that the payment equation only reflects the total cost for training job i, independent of its utility function $f_i(\cdot)$. Moreover, according to Definition 1, the payoff equals its valuation minus the payment, which has no relationship with utility function $f_i(\cdot)$. So jobs' owners would not misreport their utility functions.

Truthfulness in the required data size D_i^r and epochs E_i: Regardless of the required data sizes or epochs, as long as user i reports a larger one, it will cause its job to take longer time to compete. On the other hand, a smaller one will harm the interests of users because their ML models fail to achieve the predetermined performance, i.e., accuracy threshold. Thus, users have no motivation to misreport the required data size and epochs.

Truthfulness in the resource configuration of one worker w_i^k and PS s_i^k: Similarly, users have no motivation to report the amount of any type of resources occupied by one worker or PS, since it will reduce the computation capacity of worker/PS so that prolong the completion time of jobs. Intuitively, if users can report the resource configuration of workers/PSs with more computing power, i.e., larger w_i^k or s_i^k, their jobs can be finished more quickly. In fact, there is no need for users to do this because the more resources occupied also means that users have to pay more money to the operator. Thus, the truthfulness of the resource configuration of workers/PSs can be verified.

As a result, the proposed auction framework *Eris* can achieve truthfulness. □

Theorem 4: Algorithm 2 achieves an optimal solution of original problem (7), in which Algorithm 3 returns an optimal solution of problem (8).

Proof. Optimality of Algorithm 3: First, we try to analyze the optimality of Algorithm 3. Here we denote the schedule returned

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

by Algorithm 3 as $l_{t,d}$, which consists of data redistribution, and the deployment of workers and the PS. Note that $\lceil \frac{E_i D_i(t)}{P_i} \rceil$ represents the minimum number of workers, which can handle workload d at time t for job i. Assume that there exists one another schedule $\hat{l}_{t,d}$, which receives a smaller cost than $l_{t,d}$. Likewise, the data transmission, and the deployment of workers and the PS are indicated by $\hat{h}_i^{rr'}(t), \forall r, r' \in \mathcal{R}$, $\hat{g}_i^r(t), \forall r \in \mathcal{R}$ and $\hat{z}_i^r(t), \forall r \in \mathcal{R}$, respectively. Note that the total cost consists of data transmission cost, the resource cost of workers and the PS, and parameter exchanged cost.

We first discuss the data transmission based on a metric $\Omega_i^{rr'}(t)$. Recall that the third term and the last term can ensure the amount of transmitted data is minimum from the analysis of (10). There are two cases of this data re-distribution if necessary.

1) If $\sum_{r,r' \in \mathcal{R}} \hat{h}_i^{rr'}(t) = \sum_{r,r' \in \mathcal{R}} h_i^{rr'}(t)$, then in $\hat{l}_{t,d}$, there exists some $\hat{h}_i^{rr'}(t)$, which not equals $h_i^{rr'}(t)$. Because the amount of transmitted data remains the same, $l_{t,d}$ can be converted into $\hat{l}_{t,d}$ by transmitting data from other sites. Based on the metric $\Omega_i^{rr'}(t)$, we greedily transmit data from the site with the highest resource cost to the one with the lowest cost. So transmitting data from other sites in $\hat{l}_{t,d}$ can only increase the data transmission cost.
2) If $\sum_{r,r' \in \mathcal{R}} \hat{h}_i^{rr'}(t) > \sum_{r,r' \in \mathcal{R}} h_i^{rr'}(t)$, we need to transmit more data in the edge-network, which would cause to more data transmission cost. For the first case, we can know that moving data from other sites can increase the cost even if the amount of transmitted data is the same. So in this case, transmission cost would definitely increase when meeting more transmitted data.

Similarly, there are also two cases for the deployment of workers in schedule $l_{t,d}$. Here, we omit some details since the optimality of this part can be proved because of the similar reason. As for the deployment of the PS, deploying on the site with maximum number of workers can minimize the parameter exchanged cost. Consequently, we can conclude that *DPC_T* returns an optimal schedule $l_{t,d}, \forall t \in [a_i, \hat{t}_i], 0 \le d \le D_i$.

Optimality of Function DPC: For every completion time, we try to optimally dispatch the total workload into time slots within range $[a_i, \hat{t}_i]$ by using dynamic programming. The optimality of *DPC* can be proved by the induction on $t = a_i, \ldots, T$. At time a_i, we have $DPC(a_i, d) = DPC\_T(a_i, d), \forall 0 \le d \le D_i$, which means $DPC(a_i, d), \forall 0 \le d \le D_i$ can return the smallest cost when handling workload d. The induction hypothesis illustrates that $DPC(t, d)$ fulfills workload $d \in [0, D_i]$ with the minimum cost, for any time t. Furthermore, based on the DP function, we can get $DPC(t+1, d) = \min_{d' \in [0,d]} DPC\_T(t+1, d') + DPC(t, d-d'), \forall 0 \le d \le D_i$. Therefore, $DPC(t+1, d)$ also returns the minimum cost when fulfilling workload d at $t+1$. Consequently, we can claim that $DPC(\hat{t}_i, D_i)$ can obtain the schedule with the smallest cost at time $\hat{t}_i \in [a_i, T]$, for any jobs.

Optimality of Algorithm 2: Recall that $\hat{t}_i$ is the testing deadline rather than the final completion time. The optimal $\hat{t}_i^*$ is tight, i.e., $\sum_{t \notin [a_i, \hat{t}_i^*]} g_i^r(t) = 0$, after enumerating $\hat{t}_i$ from a_i to T. Assume that job i's final completion time is $\hat{t}_i^*$, corresponding to the optimal schedule l^*. So we can obtain the optimal schedule when $\hat{t}_i = \hat{t}_i^*$. In Algorithm 2, we enumerate $\hat{t}_i$ from a_i to T and update the optimal schedule until receiving one new schedule with smaller payoff. Recall that the utility function of jobs is non-increase with $\hat{t}_i - a_i$. In this case, Algorithm 2 would not update the optimal schedule even if there exists another schedule with the same cost and a larger deadline. So we can record the optimal schedule if we get it. Therefore, we finish this part of proof. □

Theorem 5: The competitive ratio is the upper-bound ratio of the optimal objective value to the objective value of the solution found by *Eris*. For problem (2), *Eris* is γ-competitive, where $\gamma = 2\log(\gamma_d \gamma_k^*)$, $\gamma_k^* = \max_k\{\gamma_k\}$.

Proof: Let *OPT* indicates problem (3)'s optimal objective value, which is equivalent to problem (2). Γ_i and Λ_i represent the objective value of problem (3) and (4) returned by Algorithm 1 for scheduling job i, respectively. In particular, Γ_0 and Λ_0 denote their original objective values when no job arrives. So the final objective values of the two problems are indicated by Γ_I and Λ_I. For clarify, let us introduce one lemma.

Lemma 1: When a constant γ satisfies $\Gamma_i - \Gamma_{i-1} \ge \frac{1}{\gamma}(\Lambda_i - \Lambda_{i-1}), \forall i \in \mathcal{I}$, and if $\Gamma_0 = \Lambda_0 = 0$, then Algorithm 1 is γ-competitive in total social welfare.

Proof. Given Γ_I, we can get $\Gamma_I - \Gamma_0 = \sum_{i \in \mathcal{I}} \Gamma_i - \Gamma_{i-1}$. Likewise, we have $\Lambda_I - \Lambda_0 = \sum_{i \in \mathcal{I}} \Lambda_i - \Lambda_{i-1}$. Then we can derive that $\Gamma_I - \Gamma_0 = \sum_{i \in \mathcal{I}} (\Gamma_i - \Gamma_{i-1}) \ge \frac{1}{\gamma} \sum_{i \in \mathcal{I}} (\Lambda_i - \Lambda_{i-1}) = \frac{1}{\gamma}(\Lambda_I - \Lambda_0)$. Moreover, we have $\Lambda_I \ge OPT \ge \Gamma_I$ according to the weak duality. So we have $\Gamma_I \ge \frac{1}{\gamma}\Lambda_I \ge \frac{1}{\gamma} OPT$ combining $\Gamma_0 = \Lambda_0 = 0$. Therefore, we finish this proof. □

Next we discuss how to design a specific γ to make Algorithm 1 satisfy $\Gamma_i - \Gamma_{i-1} \ge \frac{1}{\gamma}(\Lambda_i - \Lambda_{i-1}), \forall i$. There are two cases according to whether the job is accepted or not. The inequality holds since $\Gamma_i - \Gamma_{i-1} = \Lambda_i - \Lambda_{i-1} = 0$ when job i is rejected. Next, we focus on the other case where job i is accepted with its schedule l. Here, we introduce some new variables. Let $\alpha^i(t)$ indicate the unit price of data transmission at time t after accepting job i. $\beta_r^{k,i}(t)$ means the unit price of the type-k resource of site r at t after processing job i. We have

$$\begin{aligned}\alpha^i(t) - \alpha^{i-1}(t) &= (\gamma_d)^{\frac{\rho^{i-1}(t) + \psi_{il}(t)}{B}} - (\gamma_d)^{\frac{\rho^{i-1}(t)}{B}} \\ &= (\gamma_d)^{\frac{\rho^{i-1}(t)}{B}} (2^{\log(\gamma_d)^{\frac{\psi_{il}(t)}{B}}} - 1).\end{aligned}$$

Then the formula $\alpha^i(t) - \alpha^{i-1}(t) \le (\alpha^{i-1}(t) + 1)\log(\gamma_d)^{\frac{\psi_{il}(t)}{B}}$ holds by using an inequality $2^x - 1 \le x$, for any $x \in [0, 1]$. Similarly, we can get $\beta_r^{k,i}(t) - \beta_r^{k,i-1}(t) \le (\beta_r^{k,i-1}(t) + 1)\log(\gamma_k)^{\frac{\varphi_{il}^{r,k}(t)}{C_r^k}}$. Since $x_{il} = 1$, according to Algorithm 1, constraint (4a) can be guaranteed. The increased value of problem (4)'s objective is

$$\begin{aligned}\Lambda_i - \Lambda_{i-1} &= \mu_i + \sum_{r \in \mathcal{R}} \sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{T}} C_r^k (\beta_r^{k,i}(t) - \beta_r^{k,i-1}(t)) \\ &\quad + \sum_{t \in \mathcal{T}} B(\alpha^i(t) - \alpha^{i-1}(t)) \\ &\le f_i(\hat{t}_{il} - a_i) - \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}} \sum_{k \in \mathcal{K}} \varphi_{il}^{r,k}(t) \beta_r^{k,i-1}(t)\end{aligned}$$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

$$
\begin{aligned}
&- \sum_{t\in\mathcal{T}} \psi_{il}(t)\alpha^{i-1}(t) + \sum_{t\in\mathcal{T}} B(\alpha^{i-1}(t)+1)\log(\gamma_d)^{\frac{\psi_{il}(t)}{B}} \\
&+ \sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} C_r^k(\beta_r^{k,i-1}(t)+1)\log(\gamma_k)^{\frac{\varphi_{il}^{r,k}(t)}{C_r^k}} \\
&\le f_i(\hat{t}_{il}-a_i) + \log(\gamma_k^*)\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}}\varphi_{il}^{r,k}(t) \\
&+ (\log(\gamma_k^*)-1)\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}}\varphi_{il}^{r,k}(t)\beta_r^{k,i-1}(t) \\
&+ \log(\gamma_d)\sum_{t\in\mathcal{T}}\psi_{il}(t) + (\log(\gamma_d)-1)\sum_{t\in\mathcal{T}}\psi_{il}(t)\alpha^{i-1}(t),
\end{aligned}
$$

where $\gamma_k^* = \max_k\{\gamma_k\}$. According to $f_i(\hat{t}_{il}-a_i) - \sum_{t\in\mathcal{T}}\psi_{il}(t)\alpha^{i-1}(t) - \sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}\varphi_{il}^{r,k}(t)\beta_r^{k,i-1}(t) = \mu_i \ge 0$, then we have $f_i(\hat{t}_{il}-a_i) \ge \sum_{t\in\mathcal{T}}\psi_{il}(t)\alpha^{i-1}(t)$ and $f_i(\hat{t}_{il}-a_i) \ge \sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}\varphi_{il}^{r,k}(t)\beta_r^{k,i-1}(t)$. Combining aforementioned assumptions $\frac{f_i(\hat{t}_{il}-a_i)}{\sum_{t\in\mathcal{T}}\psi_{il}(t)} \ge 1$ and $\frac{f_i(\hat{t}_{il}-a_i)}{\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}}\varphi_{il}^{r,k}(t)} \ge 1$, therefore we have $\Lambda_i - \Lambda_{i-1} \le 2(\log(\gamma_d)+\log(\gamma_k^*))f_i(\hat{t}_{il}-a_i)$. In particular, $\Gamma_i - \Gamma_{i-1} = f_i(\hat{t}_{il}-a_i)$. Consequently, combining Lemma 1, we can conclude that Algorithm 1 is γ-competitive, where $\gamma = 2\log(\gamma_d\gamma_k^*)$ and $\gamma_k^* = \max_k\{\gamma_k\}$. □

Theorem 6: The time complexity of *Eris* is $O(ITD_i(R^2K + D_i))$, which is pseudo-polynomial.

Proof: To start with, we evaluate Algorithm 3's time complexity. Sorting pairs of sites in line 2 takes $O(R\log R)$ times. In lines 3–11, the *for* loop takes $O(R^2K)$ steps to calculate $h_i^{r_j r_j'}(t)$ for each pair. Similarly, sorting sites in line 12 of Algorithm 2 need $O(RK\log R)$ steps. Then, calculating the number of workers for each site and determining the placement of the PS take $O(RK)$ times at most. Thus, the time complexity of solving several one-shot problems (8) using Algorithm 3 is $O(D_iR^2K)$. In function *DPC*, we can see that the number of states (t,d) of job i is $O(TD_i)$. Furthermore, function *DPC* will be executed $O(D_i)$ times if we record intermediate values $DPC\_T(t,d)$ and $DPC(t,d)$ for all $t\in\mathcal{T}$, $d\in[0,D_i]$ to omit any extra calculations. Therefore, function $DPC(\hat{t}_i, D_i)$ needs to be executed $O(TD_i^2)$ times. In lines 2–7 of Algorithm 2, the time complexity of enumerating $\hat{t}_i$ for each arriving job is $O(TD_iR^2K + TD_i^2)$. Thus, making decisions for deploying all ML jobs of Algorithm 1 takes $O(ITD_iR^2K + ITD_i^2)$ times. Overall, the time complexity of *Eris* is $O(ITD_i(R^2K + D_i))$. □

VI. Implementation and Evaluation

In this section, we present testbed implementation in Section VI-A, and then describe testbed setup, simulator and three baselines in Section VI-B. The performance of *Eris* is presented in Section VI-C.

A. Implementation

We apply Kubernetes 1.19 to implement a distributed ML system based on Apache MXNet. Fig. 2 illustrates the testbed's overall architecture. The implemented testbed consists of nine physical machines. Each machine is equipped with 12 CPU cores, 1 GeForce RTX 2060 GPU, 16 GB RAM, 500 GB HDDs and a dual-port 1GbE NIC. In particular, we implement another physical machine as the central scheduler (Master Node). A shared Hadoop Distributed File System (HDFS) across machines will store and transmit all training data and corresponding parameters. Note that data chunk's default size is 2 MB, i.e., $\theta = 2$ MB. In each time slot, we will checkpoint and store the model parameters in HDFS [11], [26] to catch the deployment variation throughout the training process. Therefore, for training continuously, all workers need to reload the recent time slot checkpointed model parameters from HDFS.

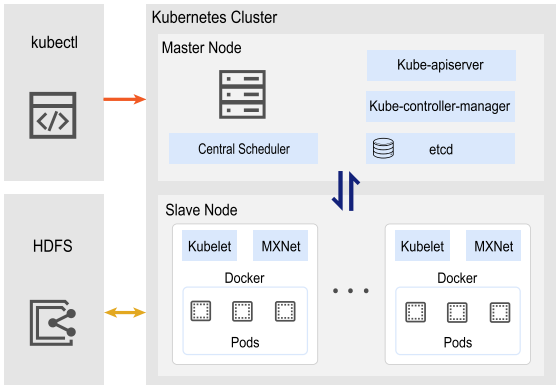

Fig. 2. Implement of the testbed.

TABLE II
DL Jobs Used for Experiments

Model	Dataset	Samples / Data chunks	Batch size / Tensor size (MB)
ResNet-50 [44]	CIFAR10	60000 / 27	64 / 97.4
ResNet-101	CIFAR10	60000 / 27	64 / 169.9
GoogLeNet [45]	Caltech101	9145 / 115	128 / 25.9
LeNet [46]	Caltech101	9145 / 115	128 / 283
AlexNet	Tiny-ImageNet [47]	100000 / 150	128 / 237.9
Inception-BN	Tiny-ImageNet	100000 / 150	128 / 42.9

Testbed Setup: Six types of DL jobs are presented in Table II. In addition, each worker or PS of jobs is implemented on a Docker container. The resources of each worker/PS are randomly chosen within the integer set [0,1] GPU, $[1,3]$ CPU cores and [2,6] RAM. The number of epochs E_i for jobs will be set to 20 or 30. In addition, we can obtain the processing capacity of workers (P_i) through pre-training. The time span of each experiment is 50 time slots, each is set to 20 minutes. In the experiments, we assume that there are 10 jobs, which will arrive randomly in the first 40 time slots. Typically, each job takes 2 to 4 hours to run. Other settings are consistent with the settings in the following simulator.

B. Experimental Setup

Simulator: We design a MATLAB simulator to evaluate *Eris*'s effectiveness at a larger scale setting by leveraging jobs' traces from our testbed. By default, there are 50 (R) sites. And 100 (I)

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

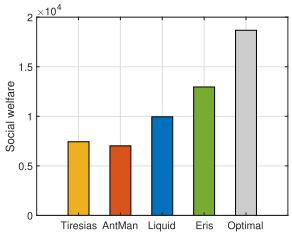

Fig. 3. Total social welfare.

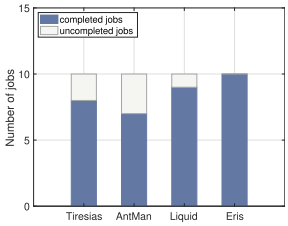

Fig. 4. # of submitted jobs.

jobs randomly arrive in 150 (T) time slots. We simulate all job events in simulator, including job arrival, completion time and corresponding resource occupation. To cater the large scale, we increase the resource configuration of each site. We set resource configuration of each site as following: 24 CPUs cores, 4 GPUs and 32 GB RAM. The overall budget of data transmission per slot is set to 20 GB. Then we set the utility function as $f_i(\hat{t}_i - a_i) = \frac{3000\tau_i}{1+e^{(\hat{t}_i - a_i)/3}}$ [24], where τ_i is a latency-sensitive factor in [1,2]. Accordingly, we set the value of γ_d and γ_k to 1000, i.e., the price functions are set as $\alpha(\rho(t)) = 1000^{\frac{\rho(t)}{B}} - 1$ and $\beta_r^k(\eta_r^k(t)) = 1000^{\frac{\eta_r^k(t)}{C_r^k}} - 1$.

Baselines: To evaluate the superiority of *Eris*, we implement three following representative schedulers:

- *Tiresias [11]:* a preemptive scheduler attempts to minimize average Job completion time. *Tiresias* allocates resources for jobs based on the number of resources, (e.g., GPUs) and the multiplication of the job's remaining workload.
- *AntMan [22]:* a cluster scheduler, which introduces two types of jobs: opportunistic job and resource-guarantee job. AntMan schedules resource-guarantee jobs first and allocates sufficient GPU resources to them. For opportunistic jobs, AntMan aims to utilize free resources to the best of its ability. Resource-guarantee jobs that suffer long queuing delay will be automatically executed as opportunistic jobs.
- *Liquid [23]:* a cluster network-efficient scheduler. Liquid tends to use the network communication cost and resource scheduling plans to obtain the best trade-off, that is, $\min \lambda * cost_k + (1-\lambda) * \sum_j scheduling\_plan_j$, where $scheduling\_plan_j$ mainly indicates the number of workers used for the job.

Note that *AntMan* and *Liquid* will assign the fixed number of workers and PSs for each new arrival job according to the unit price of worker at job's arrival time a_i, i.e., $\sum_{k\in\mathcal{K}} w_i^k \beta_r^k(a_i)$. Accordingly, to reduce data transmission consumption, the training data of each new arrival job will be re-distributed evenly and greedily to sites. For *Tiresias*, there exists an extreme case that job's payoff is smaller than 0 due to frequent preemption. We adopt it by omitting those jobs with negative payoff.

C. Evaluation Results

1) Eris *in Testbed Experiments:* In the experiments, we depict the total social welfare of four algorithms and the optimal solution in Fig. 3, which indicates that *Eris* still get a good

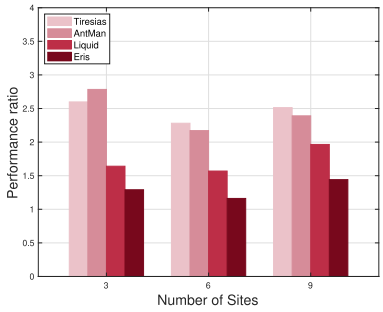

Fig. 5. Performance ratio.

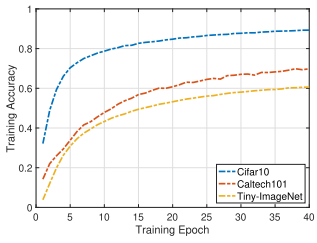

Fig. 6. Training accuracy.

performance than three baselines. Note that the optimal solution is obtained by leveraging MATLAB optimization toolbox. Besides, we use a supplementary metric (the number of rejected jobs) to assess and validate the algorithm's effectiveness. This metric provides a clear indication of the scheduling algorithms' efficiency under limited resources, elucidating how *Eris* achieves nearly-optimal social welfare. The results in Fig. 4 show that under the same resource condition, *Eris* can allocate resources for ML jobs more optimally than baselines, leading to the acceptance of more jobs (and the refusal of fewer).

The performance ratio of *Eris* with baselines are presented in Fig. 5. The performance ratio is defined as the ratio of the social welfare of the offline optimal solution to the social welfare of the target algorithm. This ratio is greater than 1, and if the ratio is near (far) 1, it means the algorithm performs well (poor) when compared to the optimal. In Fig. 5, we can see that *Eris* achieves a small performance ratio (< 1.5), which coincides with the theoretical analysis in Theorem 5.

In Fig. 6, we show the accuracy of three respective datasets. Especially, we increase the number of epochs from $20 \sim 30$ (default values) to 40 to illustrate the dynamic deployment of

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

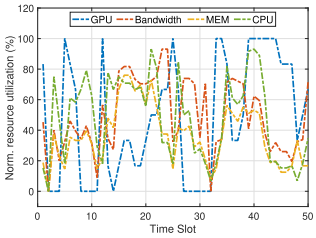

Fig. 7. Resource utilization.

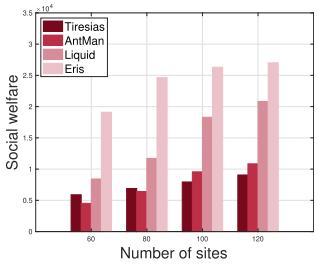

Fig. 8. Welfare under R.

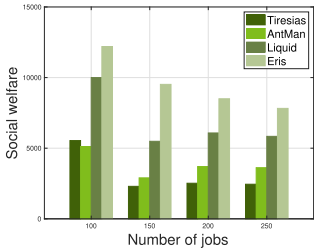

Fig. 9. Welfare under I.

workers would not affect the training process, and can still guarantee a good performance. Here, Fig. 7 demonstrates more details about the normalized resource utilization of the entire testbed in each time slot during *Eris*'s execution. Note that the normalized resource utilization represents resource utilization divided by overall testbed capacity. Especially, we depict three main resources consumed: GPU, CPU and RAM. From Fig. 7, one can see that CPU and RAM occupation are relatively stable since the price mechanism can avoid those extreme cases as mush as possible. The normalized GPU utilization sometimes reaches up to 100% because of the insufficient number of GPUs.

2) Eris *in Large-Scaled Simulations:* We evaluate the scalability of *Eris* in the case of submitting numerous jobs in a cluster with hundreds of sites. Figs. 8 and 9 show the total social welfare under different I and R, respectively. From these two figures, one can observe that *Eris* outperforms three schedulers, especially in a large scale setting. Therefore, we can claim that our algorithm *Eris* always outperforms three baselines regardless of the input scale. Furthermore, the social welfare of all algorithms in Fig. 8 increases with the number of sites as a result of increasing available resources. On the contrary, Fig. 9 shows that the social welfare of algorithms decreases with I. When the number of jobs submitted within a fixed period of time grows, it leads to a serious shortage of resources, resulting in fewer jobs completed.

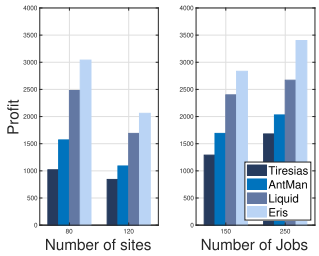

Fig. 10. Profit of the provider.

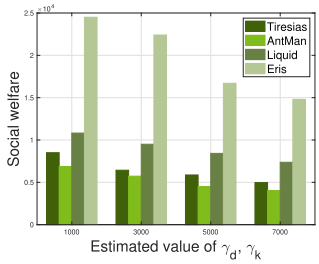

Fig. 11. Welfare under γ_d, γ_k.

In Fig. 10, we plot the service provider's profit across all algorithms. Notably, the profits decrease as the number of sites grows but rise with an increasing number of jobs. This is rationale because a provider with greater computational capacity aims to complete jobs as soon as possible to attract more users in the future, even if it means a reduced profit. Consequently, as the number of submitted jobs increases, so does the corresponding profit.

In addition, to analyze the impact of price functions, we further investigate the value of social welfare under different estimated values of γ_d, γ_k. Note that we set the value of γ_d and γ_k to be the same. In Fig. 11, we observe that higher γ_d (γ_k) leads to lower social welfare. This is because the price of unit resource $\alpha(t)$ ($\beta_r^k(t)$) increases more quickly when it meets a larger $\gamma_d(\gamma_k)$, shown in (6). In this regard, the sudden increase of price may exclude some jobs that should be accepted, leading to a smaller social welfare.

VII. Conclusion

Unbiased distributed learning (UDL) is a new emerging learning paradigm, to guarantee the quality of smart IoT services. To address the new challenges of modeling and realizing UDL, we develop an online auction-based scheduling algorithm *Eris*. *Eris* computes the best schedule for UDL jobs by using a varying number of workers and PSs such that the social welfare can be maximized, meanwhile *Eris* dynamically adjusts resource prices based on current resource consumption. This is the first work which studies the joint problem of scheduling and pricing for supporting emerged AI applications with one specific technical concern (the unbiasedness of ML models). Rigorous theoretical analysis demonstrates that *Eris* achieves a good performance ratio within polynomial running time. Our evaluations show the superiority of *Eris*, and we observe that *Eris* can improve social welfare by up to 44% compared to three representative baselines.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[2] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1521–1528.

[3] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and E. Dutkiewicz, "Optimal online data partitioning for geo-distributed machine learning in edge of wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2393–2406, Oct. 2019.

[4] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 19–25, Jan. 2020.

[5] J. Chen, K. Li, Q. Deng, K. Li, and S. Y. Philip, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Trans. Ind. Inform.*, early access, Apr. 04, 2019, doi: 10.1109/TII.2019.2909473.

[6] X. Wang, Z. Ning, and L. Wang, "Offloading in Internet of Vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018.

[7] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6722–6747, Aug. 2020.

[8] Google Cloud TPU, 2017. [Online]. Available: https://cloud.google.com/tpu

[9] Microsoft Azure machine learning, 2017. [Online]. Available: https://azure.microsoft.com/en-us/overview/machine-learning/

[10] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. 8th USENIX Conf. Netw. Syst. Des. Implementation*, 2011, pp. 323–336.

[11] J. Gu et al., "Tiresias: A GPU cluster manager for distributed deep learning," in *Proc. 16th USENIX Conf. Netw. Syst. Des. Implementation*, 2019, pp. 485–500.

[12] Y. K. Tun, Y. M. Park, N. H. Tran, W. Saad, S. R. Pandey, and C. S. Hong, "Energy-efficient resource management in UAV-assisted mobile edge computing," *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 249–253, Jan. 2021.

[13] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 360–374, Jan. 2021.

[14] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 668–682, Mar. 2019.

[15] Amazon SageMaker, 2018. [Online]. Available: https://aws.amazon.com/sagemaker/developer-resources/

[16] B. Wu, X. Chen, Y. Chen, and Y. Lu, "A truthful auction mechanism for resource allocation in mobile edge computing," in *Proc. IEEE 22nd Int. Symp. World Wireless Mobile Multimedia Netw.*, 2021, pp. 21–30.

[17] J. Lin, L. Huang, H. Zhang, X. Yang, and P. Zhao, "A novel latency-guaranteed based resource double auction for market-oriented edge computing," *Comput. Netw.*, vol. 189, 2021, Art. no. 107873.

[18] F. Mashhadi, S. A. S. Monroy, A. Bozorgchenani, and D. Tarchi, "Optimal auction for delay and energy constrained task offloading in mobile edge computing," *Comput. Netw.*, vol. 183, 2020, Art. no. 107527.

[19] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," in *Proc. ACM Int. Conf. Meas. Model. Comput. Syst.*, 2014, pp. 71–83.

[20] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 805–818, Mar. 2016.

[21] S. Li, J. Huang, and B. Cheng, "A price-incentive resource auction mechanism balancing the interests between users and cloud service provider," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 2030–2045, Jun. 2021.

[22] W. Xiao et al., "AntMan: Dynamic scaling on GPU clusters for deep learning," in *Proc. 14th USENIX Conf. Operating Syst. Des. Implementation*, 2020, Art. no. 30.

[23] R. Gu et al., "Liquid: Intelligent resource estimation and network-efficient scheduling for deep learning jobs on distributed GPU clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 2808–2820, Nov. 2022.

[24] Y. Bao, Y. Peng, C. Wu, and Z. Li, "Online job scheduling in distributed machine learning clusters," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 495–503.

[25] Q. Zhang, R. Zhou, C. Wu, L. Jiao, and Z. Li, "Online scheduling of heterogeneous distributed machine learning jobs," in *Proc. 21st Int. Symp. Theory Algorithmic Found. Protocol Des. Mobile Netw. Mobile Comput.*, 2020, pp. 111–120.

[26] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: An efficient dynamic resource scheduler for deep learning clusters," in *Proc. 13th EuroSys Conf.*, 2018, Art. no. 3.

[27] R. Zhou et al., "Online scheduling algorithm for heterogeneous distributed machine learning jobs," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1514–1529, Second Quarter 2023.

[28] R. Zhou, N. Wang, Y. Huang, J. Pang, and H. Chen, "DPS: Dynamic pricing and scheduling for distributed machine learning jobs in edge-cloud networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6377–6393, Nov. 2023.

[29] J. Pang, Z. Han, R. Zhou, H. Tan, and Y. Cao, "Online scheduling algorithms for unbiased distributed learning over wireless edge networks," *J. Syst. Archit.*, vol. 131, 2022, Art. no. 102673.

[30] J. Zhang, X. Yang, N. Xie, X. Zhang, A. V. Vasilakos, and W. Li, "An online auction mechanism for time-varying multidimensional resource allocation in clouds," *Future Gener. Comput. Syst.*, vol. 111, pp. 27–38, 2020.

[31] J. Pang, J. Yu, R. Zhou, and J. C. Lui, "An incentive auction for heterogeneous client selection in federated learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5733–5750, Oct. 2023.

[32] R. Zhou, J. Pang, Z. Wang, J. C. Lui, and Z. Li, "A truthful procurement auction for incentivizing heterogeneous clients in federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst.*, 2021, pp. 183–193.

[33] Z. Han, R. Zhou, J. Pang, Y. Cao, and H. Tan, "Online scheduling unbiased distributed learning over wireless edge networks," in *Proc. IEEE 27th Int. Conf. Parallel Distrib. Syst.*, 2021, pp. 599–606.

[34] L. Chen, W. Cui, B. Li, and B. Li, "Optimizing coflow completion times with utility max-min fairness," in *Proc. IEEE Conf. Comput. Commun.*, 2016, pp. 1–9.

[35] Z. Huang, B. Balasubramanian, M. Wang, T. Lan, M. Chiang, and D. H. Tsang, "Need for speed: CORA scheduler for optimizing completion-times in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 891–899.

[36] M. Li et al., "Scaling distributed machine learning with the parameter server," in *Proc. 11th USENIX Conf. Operating Syst. Des. Implementation*, 2014, pp. 583–598.

[37] I. Cano, M. Weimer, D. Mahajan, C. Curino, and G. M. Fumarola, "Towards geo-distributed machine learning," 2016, *arXiv:1603.09035*.

[38] F. Yan, O. Ruwase, Y. He, and T. Chilimbi, "Performance modeling and scalability optimization of distributed deep learning systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1355–1364.

[39] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, "FireCaffe: Near-linear acceleration of deep neural network training on compute clusters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2592–2600.

[40] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, pp. 2, 2012.

[41] A. Schrijver, *Theory of Linear and Integer Programming*. Hoboken, NJ, USA: Wiley, 1998.

[42] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Berlin, Germany: Springer, 1972, pp. 85–103.

[43] Z. Huang and A. Kim, "Welfare maximization with production costs: A primal dual approach," in *Proc. 26th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2015, pp. 59–72.

[44] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5987–5995.

[45] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[47] Tiny ImageNet visual recognition challenge, 2019. [Online]. Available: http://tiny-imagenet.herokuapp.com/

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.

Jinlong Pang received the BE degree from the School of Power and Machinery, and the second BE degree from the School of Computer both from Wuhan University, China, in 2019, and the ME degree from the School of Cyber Science and Engineering, Wuhan University, in 2022. Currently, he is working toward the PhD degree with the Department of Computer Science & Engineering, University of California, Santa Cruz. His current research interests include network economics, incentive mechanisms and algorithm optimization, both in the context of machine learning, especially distributed machine learning and federated learning.

Ziyi Han received the BE degree from the School of Cyber Science and Engineering, Wuhan University, China, in 2021. She is currently working toward the master's degree with the School of Cyber Science and Engineering, Wuhan University. Her research interests include edge computing, online learning, and network optimization.

Ruiting Zhou (Member, IEEE) received the PhD degree from the Department of Computer Science, University of Calgary, Canada, in 2018. She is an associate professor with the School of Computer Science Engineering, Southeast University. Her research interests include cloud computing, machine learning, and mobile network optimization. She has published research papers in top-tier computer science conferences and journals, including IEEE INFOCOM, ACM MOBIHOC, *IEEE/ACM Transactions on Networking*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*. She serves as the TPC chair for INFOCOM workshop-ICCN 2019-2023. She also serves as a reviewer for international conferences and journals, such us IEEE ICDCS, IEEE/ACM IWQoS, IEEE SECON, *IEEE Journal on Selected Areas in Communications*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Mobile Computing*, and *IEEE Transactions on Cloud Computing*.

Renli Zhang received the BE degree in information security from Wuhan University, China, in 2020. He is currently working toward the MS degree with the School of Cyber Science and Engineering, Wuhan University. His research interests include UAV-enabled wireless networks, network optimization, and online scheduling.

John C.S. Lui (Fellow, IEEE) received the PhD degree in computer science from UCLA. He is currently the Choh-Ming Li chair professor with the Department of Computer Science and Engineering (CSE), The Chinese University of Hong Kong (CUHK). After his graduation, he joined the IBM Laboratory and participated in research and development projects on file systems and parallel I/O architectures. He later joined the CSE Department with CUHK. His current research interests are in online learning algorithms and applications (e.g., multi-armed bandits, reinforcement learning), machine learning on network sciences and networking systems, large scale data analytics, network/system security, network economics, large scale storage systems, and performance evaluation theory. He is an elected member of the IFIP WG 7.3, fellow of ACM, senior research fellow of the Croucher Foundation, fellow of the Hong Kong Academy of Engineering Sciences (HKAES).

Hao Chen received the PhD degree in electronic science and engineering from Southeast University, in 2012. Currently, he is a senior engineer with Huawei Nanjing Research Institute, Nanjing, China. His research interests include storage network and resource management in cloud computing environment.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 01:02:37 UTC from IEEE Xplore. Restrictions apply.