Optimizing Resource Allocation for Joint AI Model Training and Task Inference in Edge Intelligence Systems

Xian Li, *Member, IEEE*, Suzhi Bi, *Senior Member, IEEE*, and Hui Wang

***Abstract*—This letter considers an edge intelligence system where multiple end users (EUs) collaboratively train an artificial intelligence (AI) model under the coordination of an edge server (ES) and the ES in return assists the AI inference task computation of EUs. Aiming at minimizing the energy consumption and execution latency of the EUs, we jointly consider the model training and task inference processes to optimize the local CPU frequency and task splitting ratio (i.e., the portion of task executed at the ES) of each EU, and the system bandwidth allocation. In particular, each task splitting ratio is correlated to a binary decision that represents whether downloading the trained AI model for local task inference. The problem is formulated as a hard mixed integer non-linear programming (MINLP). To tackle the combinatorial binary decisions, we propose a decomposition-oriented method by leveraging the ADMM (alternating direction method of multipliers) technique, whereby the primal MINLP problem is decomposed into multiple parallel sub-problems that can be efficiently handled. The proposed method enjoys linear complexity with the network size and simulation results show that it achieves near-optimal performance (less than 3.18% optimality gap), which significantly outperforms the considered benchmark algorithms.**

***Index Terms*—Edge intelligence, distributed training, resource allocation, alternating direction method of multipliers.**

I. Introduction

As a seamless integration of mobile edge computing (MEC) and artificial intelligence (AI), edge intelligence (EI) has grabbed the limelight from both the academia and industry [1]. Via pushing AI model training and inference towards network edges, EI is widely recognized as a promising technology to enable various computation-intensive, latency-critical, and privacy-sensitive mobile AI applications [2].

With the recent advance in distributed training techniques in EI system (e.g., federated learning [3]), end users (EUs) collaboratively train the parameters of a common AI model under the coordination of an edge server (ES). Rather than aggregating all the raw training data to a center unit, distributed learning technique allows the EUs and ES to exchange only the AI model parameters during training, which preserves EUs privacy and avoids prohibitive communication cost on massive raw data delivering. Once the training is complete, the up-to-date AI model can be used for processing the corresponding inference tasks of the EUs, e.g., image recognition. In particular, it has been shown in extensive studies that the EUs can benefit from parallel executions of computation-intensive inference tasks via task-splitting among EUs and ES, i.e., following the partial computation offloading policy [4]. In this case, the existing studies (e.g., [4]–[6]) mainly focus on jointly optimizing the task splitting ratios and system resource allocation to enhance the computation performance.

The above works share a common yet implicit assumption that the service program is always available upon task execution at all computing devices (ES and the EUs). However, in an EI system under dynamic computing environment, to avoid model degradation over time, the AI models often require to be regularly retrained and updated either periodically when sufficient new data samples are gathered at the EUs or upon significant change of environment [7]. As a result, some tasks that require high inference accuracy (e.g., recognizing images generated in new settings) can be processed only after the recent training process is complete. Besides the causal relationship, the training and inference processes share the common system resource (e.g., limited bandwidth for model/task data transmission), thus should be jointly treated to minimize the overall cost. In a multiuser EI system, existing studies on resource allocation mostly investigate the training (e.g., [8], [9]) and inference processes (e.g., [4]–[6], [10], [11]) separately, while the joint design problem, to the best of our knowledge, is currently lacking of concrete study. The major difficulty lies in the resource sharing not only between the dependent training and inference processes, but also among the operations of interdependent and heterogeneous EUs in each process, featured by their dissimilar inference task volume, hardware capabilities, and wireless channel conditions, etc.

In this letter, we study the optimal resource allocation problem for joint AI model training and task inference in a multi-user EI system. In particular, we consider that the system conducts on-device federated learning to iteratively train an AI model, and later uses the trained model to process inference tasks of the EUs. We aim to minimize the weighted summation of energy consumption and execution latency (WSEL) of all EUs in completing their training/computation tasks. The problem involves optimizing not only the computation task splitting ratio of each individual EU, but also the system-level bandwidth allocation on transmitting the AI model and user task data. In particular, each task splitting ratio is correlated to a binary decision that represents whether downloading the trained AI model for local task inference. The problem is formulated as a hard MINLP, where we handle the intractability by an efficient decomposition-oriented method leveraging the ADMM (alternating direction method of multipliers) technique. Finally, we conduct numerical simulations and show that the proposed joint optimization achieves a near-optimal

Manuscript received August 31, 2020; accepted November 5, 2020. Date of publication November 10, 2020; date of current version March 9, 2021. This work was supported in part by the National Key Research and Development Program under Project 2019YFB1803305; in part by the National Natural Science Foundation of China under Project 61871271; in part by the Guangdong Province Pearl River Scholar Funding Scheme 2018; in part by the Key Project of Department of Education of Guangdong Province under Grant 2020ZDZX3050; and in part by the Foundation of Shenzhen City under Project JCYJ20170818101824392 and Project JCYJ20190808120415286. The associate editor coordinating the review of this article and approving it for publication was G. Zheng. *(Corresponding authors: Hui Wang; Suzhi Bi.)*

Xian Li and Hui Wang are with the College of Electronic and Information Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: xianli@szu.edu.cn; wanghsz@szu.edu.cn).

Suzhi Bi is with the College of Electronic and Information Engineering, Shenzhen University, Shenzhen 518060, China, and also with Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: bsz@szu.edu.cn).

Digital Object Identifier 10.1109/LWC.2020.3036852

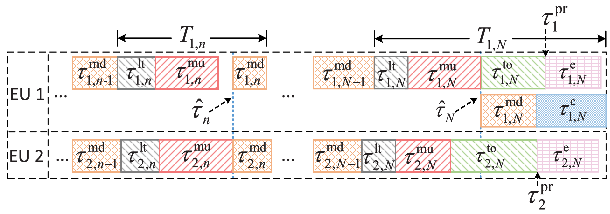

Fig. 1. An example of the system operation.

performance that considerably reduces the WSEL compared to the considered benchmark algorithms, e.g., allocating dedicated bandwidth for AI model transmissions.

II. System Model and Problem Formulation

A. System Model

In this letter, we consider an EI system composing of one ES and M EUs that regularly update an AI model for executing a certain type of computation task. Let $\mathcal{M}$ be the set of EUs. Upon receiving the model update request from the ES, the EUs first download the most recent AI model from the ES, and then collaboratively retrain the AI model following the federated learning approach through N fixed training iterations. In each training iteration, EUs process their local training samples and transmit to the ES their local model parameter updates (MPUs). The ES aggregates the received MPUs into a new AI model and transmits it back to EUs for generating MPUs in the next iteration [9]. To achieve high inference accuracy, we consider that the EUs perform only model training in the first $N-1$ iterations, while both training and inference in the N-th iteration, such that the inference tasks are processed by the most up-to-date AI model. For the task inference in the N-th iteration, we adopt a partial offloading policy that an EU can arbitrarily partition its inference task data with one part computed locally and the other offloaded to the ES for edge computation [4].[1] Notice that an EU needs to download the AI model only if it processes some of its task locally, while those fully rely on the ES for edge inference do not. To avoid co-channel interference, the EUs are allocated with orthogonal sub-channels in the uplink and downlink communications. With separate circuits, the EUs and the ES can perform task computation and data communication simultaneously.

An example operation of the training and inference process with $M = 2$ is shown in Fig. 1, where the first EU has its task processed both locally and at the ES, while the second one performs edge inference only. In the n-th ($1 \leq n < N$) iteration, both EUs first perform local model training, followed by local MPU uploading, where the delays are denoted by $\tau_{j,n}^{\text{lt}}$ and $\tau_{j,n}^{\text{mu}}$, respectively. We assume the local model training is operated over a preset numbers of local iterations with a fixed CPU frequency, such that the delay $\tau_{j,n}^{\text{lt}}$ and energy consumption (denoted as $e_{j,n}^{\text{lt}}$) are known constant. We assume that the ES adopts the synchronous model aggregation approach [12], which updates the AI model after receiving the MPUs from both EUs. For simplicity, we neglect the model aggregation delay and denote the time that the ES finishes updating the AI model as $\hat{\tau}_n$. Then, each EU downloads the updated model for a duration denoted by $\tau_{j,n}^{\text{md}}$, and retrains the model in the $(n+1)$-th iteration. In the N-th iteration, after local training and MPU uploading, the two EUs start offloading their task data to the ES for edge inference, taking $\tau_{1,N}^{\text{to}}$ and $\tau_{2,N}^{\text{to}}$ amount of time, respectively. The first EU starts downloading the model from time instant $\hat{\tau}_N$ for a duration $\tau_{1,N}^{\text{md}}$, with which it performs local inference with duration $\tau_{1,N}^{\text{c}}$. Meanwhile, the ES starts computing the task of EU j for a duration of $\tau_{j,N}^{\text{e}}$ once its task data is completely received (e.g., from the time instant τ_j^{pr} for EU j), where $j = 1, 2$.

In this letter, we optimize the system performance in each iteration. We first focus on formulating the resource allocation problem in the N-th iteration and later show that the problem in iteration $n \in [1, N-1]$ is a special case.

[1]It is worth noting that the privacy of the training data at users is generally not affected by the following inference process. For simplicity, we assume that the inference task data does not contain privacy-sensitive contents, such that an arbitrary part of the task can be offloaded.

B. Problem Formulation

We denote the task data size of the j-th EU as L_j in bits, where $\iota_j L_j$-bits ($0 \leq \iota_j \leq 1$) data is offloaded to the ES for edge computing, and the rest $(1-\iota_j)L_j$-bits data is computed locally. Denote C as the required CPU cycles to process one bit of data. Then, the computation time and the corresponding energy consumption of local computing can be given as [5]

$$\tau_{j,N}^{\text{c}} = C(1-\iota_j)L_j/f_j,\ e_{j,N}^{\text{c}} = \kappa_{\text{c}} f_j^2 C(1-\iota_j)L_j, \quad (1)$$

respectively, where f_j, constrained by the maximum value $f_{\max}$, is the local CPU frequency. κ_{c} is the energy efficiency for local computing. Besides, the time spent on edge computing on the ES side is

$$\tau_{j,N}^{\text{e}} = C\iota_j L_j / f_{\text{s}}, \quad (2)$$

where f_{s} is the fixed CPU frequency at the ES assigned to process a task.

In the N-th iteration, we assume that EU j is allocated with $b_j B$ dedicated bandwidth in its uplink communication, where B is the total available bandwidth. We denote p_j as the fixed transmit power at EU j for transmitting its local MPU and task data. Then, the achievable rate of the j-th UE is

$$r_j = b_j B \log_2\left[1 + p_j h_j / \left(b_j B \delta_{\text{s}}^2\right)\right], \quad (3)$$

where h_j is the channel power gain between the ES and the j-th EU, and δ_{s}^2 is the noise power density at the ES side. We assume for simplicity that the channel is reciprocal. Then, the achievable data rate for fetching the aggregated AI model from the ES to the j-th EU is

$$\bar{r}_j = \bar{b}_j B \log_2\left[1 + P_{\text{s}} h_j / \left(\bar{b}_j B \delta_j^2\right)\right], \quad (4)$$

where $\bar{b}_j B$ is the assigned downlink bandwidth. P_{s} is the fixed transmit power of the ES, and δ_{j}^2 is the noise power density.

With the above communication models, the transmission latency of MPU uploading, task offloading, and AI model downloading can be given as

$$\tau_{j,N}^{\text{mu}} = D_{\text{m}}/r_j,\quad \tau_{j,N}^{\text{to}} = \iota_j L_j / r_j,\quad \tau_{j,N}^{\text{md}} = D_{\text{m}}/\bar{r}_j, \quad (5)$$

respectively, where D_{m} is the data size of the AI model. Then, the time that the AI model is globally updated at the ES is

$$\hat{\tau}_N = \max\left\{\tau_{j,N-1}^{\text{md}} + \tau_{j,N}^{\text{lt}} + \tau_{j,N}^{\text{mu}}, \forall j \in \mathcal{M}\right\}. \quad (6)$$

Correspondingly, the energy costs on data transmissions are

$$e_{j,N}^{\text{mu}} = p_j \tau_{j,N}^{\text{mu}},\ e_{j,N}^{\text{to}} = p_j \tau_{j,N}^{\text{to}},\ e_j^{\text{md}} = p_j^{\text{md}} \tau_{j,N}^{\text{md}}, \quad (7)$$

respectively, where p_j^{md} in Joule/bit is a constant representing the receiver circuit power at the j-th EU.

We consider that the ES has stable power supply and focus only on the energy consumption of the EUs. Let $a_j \in \{0, 1\}$ be the model download decision of the j-th EU (i.e., $a_j = 1$ if the j-th EU downloads the aggregated AI model, otherwise $a_j =$

0). Since an EU downloads the model only when performing local task processing, we have that $a_j = 1$ when $0 \leq \iota_j < 1$ and $a_j = 0$ when $\iota_j = 1$. Then, the total energy consumption of an EU $j \in \mathcal{M}$ in the *N*-th iteration can be written as

$$E_{j,N} = e^{\mathrm{lt}}_{j,N} + e^{\mathrm{mu}}_{j,N} + e^{\mathrm{to}}_{j,N} + a_j\big(e^{\mathrm{md}}_{j,N} + e^{\mathrm{c}}_{j,N}\big). \tag{8}$$

Correspondingly, the execution latency is

$$T_{j,N} = \max\left\{a_j\left(\hat{\tau}_N + \tau^{\mathrm{md}}_{j,N} + \tau^{\mathrm{c}}_{j,N}\right), \tau^{\mathrm{pr}}_j + \tau^{\mathrm{e}}_{j,N}\right\} - \tau^{\mathrm{md}}_{j,N-1}, \tag{9}$$

where $\tau^{\mathrm{pr}}_j = \max\{\hat{\tau}_N, \tau^{\mathrm{md}}_{j,N-1} + \tau^{\mathrm{lt}}_{j,N} + \tau^{\mathrm{mu}}_{j,N} + \tau^{\mathrm{to}}_{j,N}\}$ denotes the time that both the AI model and task data are ready for processing the task of EU *j*. Notice that $\tau^{\mathrm{md}}_{j,N-1}$ is known from the $(N-1)$-th iteration.

In this letter, we are interested in minimizing the WSEL of the considered system, which is a commonly used metric to jointly evaluate the system performance on energy and latency cost [11]. Notice that minimizing WSEL can also alleviates the negative impact of the energy and latency cost induced by the communication overhead of federated learning during model training. Specifically, we solve

$$\min_{\substack{\boldsymbol{b},\boldsymbol{a},\boldsymbol{\iota},\boldsymbol{f},\\ \boldsymbol{\tau},\boldsymbol{T},\hat{\tau}_N}} \ \sum\nolimits_{j\in\mathcal{M}}\big(w^{\mathrm{e}}_j E_{j,N} + w^{\mathrm{t}}_j T_{j,N}\big) \tag{10a}$$

$$\text{s.t.} \ \sum\nolimits_{j\in\mathcal{M}}\big(b_j + \bar{b}_j\big) \leq 1, \quad b_j, \bar{b}_j \geq 0, \forall j \in \mathcal{M}, \tag{10b}$$

$$0 \leq f_j \leq f_{\max}, \forall j \in \mathcal{M}, \tag{10c}$$

$$\{a_j, \iota_j\} \in \mathcal{D}_j, \forall j \in \mathcal{M}, \tag{10d}$$

$$\{\hat{\tau}_N, \boldsymbol{\tau}, \boldsymbol{T}\} \in \mathcal{T}, \tag{10e}$$

where $\boldsymbol{b} = \{b_j, \bar{b}_j\}$, $\boldsymbol{a} = \{a_j\}$, $\boldsymbol{\iota} = \{\iota_j\}$, $\boldsymbol{f} = \{f_j\}$. $\boldsymbol{\tau} = \{\tau^{\mathrm{mu}}_{j,N}, \tau^{\mathrm{to}}_{j,N}, \tau^{\mathrm{pr}}_j, \tau^{\mathrm{md}}_{j,N}\}$, $\boldsymbol{T} = \{T_{j,N}\}$, and $j \in \mathcal{M}$. w^{e}_j and w^{t}_j denote the weighting factors of the energy consumption and execution delay of the *j*-th EU, respectively. (10b) is the constraints on bandwidth allocation. (10c) bounds the local CPU frequency at EUs. In (10d) and (10e), $\mathcal{D}_j$ describes the constraints on model download decision and task splitting ratio, and $\mathcal{T}$ captures the feasible region of the time costs $\{\hat{\tau}_N, \boldsymbol{\tau}, \boldsymbol{T}\}$, both of which are detailed in (11) as shown at the bottom of the page. By assigning a zero-bit task (i.e., $L_j = 0$) and enforcing model download (i.e., $a_j = 1$) for each EU, (10) turns out to be the optimization problem in the preceding training iteration $n \in [1, N-1]$. In this letter, we focus on solving (10), while the optimization problems of the first $N-1$ iterations can be tackled in a similar and in fact much easier way.

III. Proposed ADMM-Based Method

The non-convex terms (e.g., $\iota_j f_j$ and ι_j / f_j) and the binary variables a_j-s make (10) intractable. A potential method to solve (10) is to fix $\boldsymbol{\iota}$ and $\boldsymbol{a}$, then the remaining problem is convex and can be solved via off-the-shelf tools (e.g., interior point method). The optimal $\{\boldsymbol{a}, \boldsymbol{\iota}\}$ can be found via enumerating all the $\prod_{j=1}^{M}(\Delta\iota_j)^{-1}$ possible combinations, where $\Delta\iota_j$ is a sufficiently small precision interval of l_j. However, when *M* becomes large, the exhaustive search on $\{\boldsymbol{a}, \boldsymbol{\iota}\}$ is prohibitive. To deal with this problem, we propose an ADMM-based method that decomposes the large-size MINLP into parallel smaller and tractable sub-problems. We later show that it enjoys linear computation complexity *O*(*M*).

To start with, we introduce auxiliary variables $\boldsymbol{x} = \{x_j\}$, $\boldsymbol{y} = \{y_j\}$, $\boldsymbol{z} = \{z_j\}$, where $j \in \mathcal{M}$, and convert (10) into an equivalent form as below:

$$\min_{\boldsymbol{u},\hat{\tau}_N} \ \sum\nolimits_{j\in\mathcal{M}}\big(q_j(u_j) + g(\hat{\tau}_N, \boldsymbol{b})\big) \tag{12a}$$

$$\text{s.t.} \ x_j = b_j, y_j = \hat{\tau}_N, z_j = \bar{b}_j, \forall j \in \mathcal{M}, \tag{12b}$$

$$x_j, y_j, z_j \geq 0, \forall j \in \mathcal{M}, \tag{12c}$$

$$\text{(10c)-(10e)}, \tag{12d}$$

where $u_j = \{x_j, y_j, z_j, a_j, \iota_j, f_j, \tau^{\mathrm{mu}}_{j,N}, \tau^{\mathrm{to}}_{j,N}, \tau^{\mathrm{md}}_{j,N}, \tau^{\mathrm{pr}}_j, T_{j,N}\}$ and $\boldsymbol{u} = \{u_j, j \in \mathcal{M}\}$. In the objective function (12a), $q_j(u_j) = w^{\mathrm{e}}_j E_{j,N} + w^{\mathrm{t}}_j T_{j,N}$ and

$$g(\hat{\tau}_N, \boldsymbol{b}) = \begin{cases} 0, & \text{if } \{\hat{\tau}_N, \boldsymbol{b}\} \in \mathcal{F}, & \text{(13a)} \\ +\infty, & \text{otherwise}, & \text{(13b)} \end{cases}$$

where $\mathcal{F} = \{\hat{\tau}_N, \boldsymbol{b} \mid \sum_{j\in\mathcal{M}}(b_j + \bar{b}_j) \leq 1; b_j \geq 0, \bar{b}_j \geq 0, \ \forall j \in \mathcal{M}; \hat{\tau}_N \geq 0\}$. Let $\boldsymbol{\alpha} = \{\alpha_j\}$, $\boldsymbol{\beta} = \{\beta_j\}$ and $\boldsymbol{\zeta} = \{\zeta_j\}$, $\forall j \in \mathcal{M}$, be the Lagrangian multipliers associated with constraints in (12b), and *c* be a fixed updating step size. Then, we can write a partial augmented Lagrangian of (12) as

$$\begin{aligned} \boldsymbol{\mathcal{L}}_{\mathbf{A}}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\omega}) &= \sum_{j\in\mathcal{M}} q_j(u_j) + g(\boldsymbol{v}) + \sum_{j\in\mathcal{M}} \alpha_j\big(x_j - b_j\big) \\ &+ \sum_{j\in\mathcal{M}} \beta_j\big(y_j - \hat{\tau}_N\big) + \sum_{j\in\mathcal{M}} \zeta_j\big(z_j - \bar{b}_j\big) + \frac{c}{2}\sum_{j\in\mathcal{M}} \big(x_j - b_j\big)^2 \\ &+ \frac{c}{2}\sum_{j\in\mathcal{M}} \big(y_j - \hat{\tau}_N\big)^2 + \frac{c}{2}\sum_{j\in\mathcal{M}} \big(z_j - \bar{b}_j\big)^2, \end{aligned} \tag{14}$$

where $\boldsymbol{v} = \{\hat{\tau}_N, \boldsymbol{b}\}$ and $\boldsymbol{\omega} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}\}$. Correspondingly, the optimization of (12) are separated into two levels. In specific, at the lower level, the dual function $K(\boldsymbol{\omega})$ is obtained via solving the following optimization problem:

$$K(\boldsymbol{\omega}) = \min_{\boldsymbol{u},\boldsymbol{v}} \ \boldsymbol{\mathcal{L}}_{\mathbf{A}}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\omega}), \ \text{s.t.}\,(12\text{c}), (10\text{c})\text{–}(10\text{e})\,. \tag{15}$$

At the higher level, we solve the dual problem

$$\max_{\boldsymbol{\omega}\geq 0} \ K(\boldsymbol{\omega}). \tag{16}$$

By leveraging the ADMM technique, we decompose (15) into *M* parallel sub-problems and solve (16) via iteratively updating $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{\omega}$. Denote the values in the *I*-th iteration as $\{\boldsymbol{u}^I, \boldsymbol{v}^I, \boldsymbol{\omega}^I\}$. Then, in the $(I+1)$-th iteration, the ADMM-based method runs as follows:

1) *Step 1:* Given $\{\boldsymbol{v}^I, \boldsymbol{\omega}^I\}$, we first update $\boldsymbol{u}$ by solving

$$\boldsymbol{u}^{I+1} = \arg\min_{\boldsymbol{u}} \ \boldsymbol{\mathcal{L}}_{\mathbf{A}}(\boldsymbol{u}, \boldsymbol{v}^I, \boldsymbol{\omega}^I). \tag{17}$$

Let $S^I(u_j) = q_j(u_j) + \alpha^I_j x_j + \beta^I_j y_j + \zeta^I_j z_j + \frac{c}{2}[(x_j - b^I_j)^2 + (y_j - \hat{\tau}^I_N)^2 + (z_j - \bar{b}^I_j)^2]$. Then, (17) is decomposed into *M* parallel subproblems, each solving

$$u^{I+1}_j = \arg\min_{u_j\in\mathcal{U}_j} \ S^I(u_j), \ \text{s.t.} \ (12\text{c}), (10\text{c}), (10\text{d})\,. \tag{18}$$

Here, $\mathcal{U}_j$ is the feasible set of u_j, which is given by replacing $b_j = x_j$, $\hat{\tau}_N = y_j$ and $\bar{b}_j = z_j$ in $\boldsymbol{\mathcal{T}}$ as in (11). Now that a_j is given in the current step, we further consider a fixed l_j, such that (10d) is removed from (18) and the problem becomes a convex optimization problem that can be well tackled. Subsequently, we update ι_j for each EU using one-dimensional search method (e.g., the golden-section search or

$$\begin{cases} \boldsymbol{\mathcal{T}} = \Big\{T_{j,N}, \hat{\tau}_N, \tau^{\mathrm{mu}}_{j,N}, \tau^{\mathrm{to}}_{j,N}, \tau^{\mathrm{md}}_j, \tau^{\mathrm{tp}}_j, \forall j \in \mathcal{M} \mid T_{j,N} \geq a_j\Big(\hat{\tau}_N + \tau^{\mathrm{md}}_{j,N} + C(1-\iota_j)L_j/f_j\Big) - \tau^{\mathrm{md}}_{j,N-1}, \ T_{j,N} \geq \tau^{\mathrm{pr}}_j + C\iota_j L_j/f_{\mathrm{s}} - \tau^{\mathrm{md}}_{j,N-1} \\ \qquad \hat{\tau}_N \geq \tau^{\mathrm{md}}_{j,N-1} + \tau^{\mathrm{lt}}_{j,N} + \tau^{\mathrm{mu}}_{j,N}, \tau^{\mathrm{pr}}_j \geq \hat{\tau}_N, \tau^{\mathrm{pr}}_j \geq \tau^{\mathrm{md}}_{j,N-1} + \tau^{\mathrm{lt}}_{j,N} + \tau^{\mathrm{mu}}_{j,N} + \tau^{\mathrm{to}}_{j,N}, \tau^{\mathrm{mu}}_{j,N} \geq D_s/r_j, \tau^{\mathrm{to}}_{j,N} \geq \iota_j L_j/r_j, \tau^{\mathrm{md}}_{j,N} \geq D_{\mathrm{m}}/\bar{r}_j\Big\} \\ \mathcal{D}_j = \{a_j, \iota_j \mid a_j \in \{0,1\}, 0 \leq \iota_j \leq 1, (1-a_j)(1-\iota_j) = 0\} \setminus \{a_j = 1, \iota_j = 1\} \end{cases} \tag{11}$$

the data-driven-based search [13]). [2] Here, we provide below some interesting properties of the optimal solutions of (18) that can be used to simplify the algorithm design.

Proposition 1: For given ι_j, the local CPU frequency of the *j*-th EU at the optimum of (18) can be given as

$$f_j = \min\left(\sqrt[3]{\mu_j/(w_j^{\mathrm{e}} 2\kappa_{\mathrm{c}})}, f_{\max}\right), \tag{19}$$

where μ_j is a non-negative Lagrangian multiplier and $\min(\cdot)$ is the minimum operation. The corresponding execution latency of the *j*-th EU at the optimum of (18) can be given as $T_{j,N}^* = \hat{T}_{j,N}^* - \tau_{j,N-1}^{\mathrm{md}}$, where

$$\hat{T}_{j,N}^* = \begin{cases} y_j + \tau_{j,N}^{\mathrm{md}} + C(1-\iota_j)L_j/f_j, & \text{if } 0 \le \iota_j < 1, \quad (20a) \\ \tau_j^{\mathrm{pr}} + CL_j/f_{\mathrm{s}}, & \text{if } l_j = L_j. \quad (20b) \end{cases}$$

Proof: Please refer to Appendix A. ■

Proposition 1 implies that, as long as some task data is processed locally (i.e., $\iota_j < 1$), the local processing delay dominates the edge processing delay. Accordingly, we can remove the maximum operator in (9) and simplify the delay expression to (19). Notice that the complexity of solving each sub-problem (18) does not scale with the EU number, thus the overall complexity of step 1 is *O*(*M*), i.e., proportional to the number of sub-problems.

2) *Step 2:* With the obtained $\boldsymbol{u}^{I+1}$, we update $\boldsymbol{v}^{I+1}$ by solving the following convex optimization problem:

$$\min_{\{\hat{\tau}_N, \boldsymbol{b}\}\in\mathcal{F}} \sum\nolimits_{j\in\mathcal{M}} \left[\alpha_j^I \left(x_j^{I+1} - b_j\right) + \beta_j^I \left(y_j^{I+1} - \hat{\tau}_N\right) + \zeta_j^I \left(z_j^{I+1} - \bar{b}_j\right)\right]$$
$$+ \frac{c}{2}\sum\nolimits_{j\in\mathcal{M}} \left[\left(x_j^{I+1} - b_j\right)^2 + \left(y_j^{I+1} - \hat{\tau}_N\right)^2 + \left(z_j^{I+1} - \bar{b}_j\right)^2\right]. \tag{21}$$

The optimal solution is in the following closed-form:

$$\hat{\tau}_N^* = \frac{1}{M}\sum_{j\in\mathcal{M}}\left(y_j^{I+1} + \frac{\beta_j^I}{c}\right),\ b_j^* = \left(x_j^{I+1} + \frac{\alpha_j^I - \theta}{c}\right)^+,\ \bar{b}_j^* = \left(z_j^{I+1} + \frac{\zeta_j^I - \theta}{c}\right)^+, \tag{22}$$

where $j = 1, \ldots, M$ and $(\cdot)^+ \triangleq \max(\cdot, 0)$. θ^* is the optimal dual variable associated with the bandwidth allocation constraint in $\mathcal{F}$ and can be obtained via a bi-section search. The details are omitted due to the page limit. The complexity of the bi-section search method to solve (21) is *O*(*M*).

3) *Step 3:* given $\boldsymbol{u}^{I+1}$ and $\boldsymbol{v}^{I+1}$, we update the multipliers $\boldsymbol{\omega}^I$ using the sub-gradient method as below:

$$\alpha_j^{I+1} = \alpha_j^I + c\left(x_j^{I+1} - b_j^{I+1}\right),\ j = 1, \ldots, M, \tag{23a}$$
$$\beta_j^{I+1} = \beta_j^I + c\left(y_j^{I+1} - \hat{\tau}_N^{I+1}\right),\ j = 1, \ldots, M, \tag{23b}$$
$$\zeta_j^{I+1} = \zeta_j^I + c\left(z_j^{I+1} - \bar{b}_j^{I+1}\right),\ j = 1, \ldots, M. \tag{23c}$$

4) Finally, the iteration of the ADMM-based method stops when the following two criteria are satisfied:

$$\sum\nolimits_{j\in\mathcal{M}} \left(|x_j^{I+1} - b_j^{I+1}| + |y_j^{I+1} - \hat{\tau}_N^{I+1}| + |z_j^{I+1} - \bar{b}_j^{I+1}|\right) \le \epsilon_1, \tag{24a}$$
$$\sum\nolimits_{j\in\mathcal{M}} \left(|b_j^I - b_j^{I+1}| + |\bar{b}_j^I - \bar{b}_j^{I+1}|\right) + |\hat{\tau}_N^I - \hat{\tau}_N^{I+1}| \le \epsilon_2, \tag{24b}$$

where ϵ_1 and ϵ_2 denote the absolute tolerance and relative tolerance [14], respectively. Otherwise, the $(I+2)$-th iteration continues from Step 1.

The complexity of step 3, and thus that of an ADMM iteration of the first three steps is *O*(*M*). Besides, the *M* sub-problems in step 1 can be computed in parallel to reduce the computation time. Meanwhile, we observe through extensive simulations that the number of ADMM iterations required until convergence does not vary significantly with *M* (e.g., in less than 40 – 60 iterations for any $M \in [2, 12]$), where the result is omitted due to the page limit. Overall, the proposed method enjoys a linear computation complexity, i.e., *O*(*M*), which greatly reduces the computation time and makes the primal problem (10) tractable even at a large *M*.

[2] In practice, the edge processing frequency f_{s} could also be optimized to further improve the system performance, which can be achieved with a tiny modification in this step without incurring additional technical difficulty.

TABLE I
SIMULATION PARAMETERS

$P_{\mathrm{B}} = 33$ dBm	$p_{\max} = 23$ dBm	$p^{\mathrm{md}} = 0.97$ mW
$B = 20$ MHz	$k_{\mathrm{c}} = 0.5 \times 10^{-28}$	$C = 50$ cycles/bit
$\delta_{\mathrm{b}}^2 = \delta_j^2 = -174$ dBm/Hz	$f_{\mathrm{s}} = 4$ GHz	$f_{\max} = 1$ GHz
$w^{\mathrm{e}} = 0.9$	$w^{\mathrm{t}} = 0.1$	$c = 20$
$\epsilon_1 = 1.5M \times 10^{-4}$	$\epsilon_2 = M \times 10^{-4}$	$L = 64$ Mbits

IV. SIMULATION RESULTS

In this section, we evaluate the system performance via numerical simulations. We consider an EI system comprising $M = 4$ EUs, deployed around the ES at an equal distance $d_j = 50$ m. We model the channel between the *j*-th EU and the ES as a Rayleigh fading channel. Then, the corresponding channel gain is $h_j = \varsigma \bar{h}_j$. Here, ς is an independent exponential random variable of unit mean, which captures the small-scale channel fading effect. $\bar{h}_j$ denotes the average channel gain that follows a path-loss model $\bar{h}_j = G_{\mathrm{A}}(\frac{3\times10^8}{4\pi f_{\mathrm{c}} d_j})^{\sigma}$, where $G_{\mathrm{A}} = 4.11$ captures the total antenna gain, $f_{\mathrm{c}} = 2.4$ GHz represents the carrier frequency, and σ is the path-loss exponent, respectively. We assume all EUs are identical in task data bulk *L*, maximum transmit power $p_{\max}$, receiver power p^{md}, and weighting factors w^{e} and w^{t} (here we drop the subscript '*j*' for convenience). Without loss of generality, we set $\tau_{j,N-1}^{\mathrm{md}} = \tau_{j,N-1}^{\mathrm{lt}} = 1$ s and $e_{j,N-1}^{\mathrm{lt}} = 0.04$ Joule, $\forall j \in \mathcal{M}$. The other parameters used in simulation are listed in Table I.

To verify the effectiveness of the proposed method, we consider four representative benchmarks for comparison:

- *Local computing only (LCO):* all EUs execute their tasks locally, i.e., $L_j = L$, $\forall j \in \mathcal{M}$.
- *Edge computing only (ECO):* all EUs offload all their tasks to the ES (i.e., $L_j = 0$ and $a_j = 0$, $\forall j \in \mathcal{M}$).
- *Group bandwidth equalization (GBE):* The uplink and downlink transmissions occupy equal bandwidth $\frac{B}{2}$ each.
- *Optimal:* exhaustively enumerating all the $\prod_{j=1}^{M} \frac{1}{\Delta\iota_j}$ combinations of task offloading choices, where $\Delta\iota_j = \Delta\iota = 0.2, \forall j \in \mathcal{M}$.

The ADMM-based method adopts the golden-section search to update the value of ι_j, with an absolute precision $\epsilon_0 = 10^{-3}$. For all the considered benchmarks, except for the specified values, all the other variables are optimized, where the details are omitted for brevity. All the simulation results are obtained through averaging 10 channel realizations. Notice that we consider only 4 users in the simulation because the complexity in computing the optimal solution using the enumeration based method is prohibitive even when N is small, e.g., $N = 8$. Nonetheless, we observe the similar performance advantages of the proposed method when *N* is larger, which is omitted here due to the page limit.

In Fig. 2(a), we first demonstrate the WSEL performance of different methods with varying model size D_{m}. In this simulation, the path-loss exponent is set as $\sigma = 2.9$. We also evaluate the impact of $\Delta\iota$ on the performance of exhaustive searching by considering $\Delta\iota = 0.5, 0.2, 0.1$, respectively. The results show that the performance of exhaustive searching

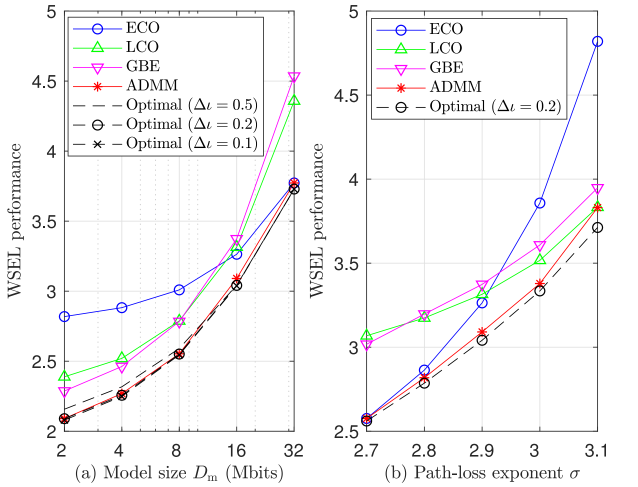

Fig. 2. (a) The WSEL performance versus model size $D_{\rm m}$. (b) The WSEL performance versus path-loss exponent σ.

is improved when $\Delta\iota$ varies from 0.5 to 0.2, while the improvement becomes marginal when using $\Delta\iota = 0.1$. We therefore use $\Delta\iota = 0.2$ in the following simulation. For the GBE method, the WSEL performance is better than ECO and LCO at a small $D_{\rm m}$, whereas dramatically deteriorates when $D_{\rm m}$ becomes large. Besides, a crossover can be observed between ECO and LCO with the rise of $D_{\rm m}$. Eventually, ECO achieves a similar grade to the ADMM-based method. This is because that EUs tend to perform edge computing at a large $D_{\rm m}$. Nevertheless, the ADMM-based method shows a significant superiority over these three benchmarks and achieves near-optimal performance for all values of $D_{\rm m}$.

Then, the WSEL performance of the system is depicted in Fig. 2(b) as a function of the path-loss exponent σ with $D_{\rm m} = 16$ Mbits. It displays that the ADMM-based method provides significant performance improvement compared with the benchmarks and achieve near-optimal performance (less than 3.18% optimality gap) for all considered σ-s. For the GBE method, it gets lower WSEL at small σ-s (e.g., $\sigma = 2.7$) than LCO and performs better than ECO when σ is large. The ECO method can approach the performance of ADMM at small σ, but causes drastic performance degradation with the growth of σ. On the contrary, the LCO method shows a different trend, where its performance gap between the ADMM-based method shrinks as σ increases. This is because a larger σ leads to a severer signal attenuation due to path-loss and thus a higher cost for data offloading.

V. Conclusion

This letter studied the optimal resource allocation problem for joint computation task processing and distributed training in a multi-user edge intelligence network. We formulated an MINLP problem to minimize the computation delay and energy consumption of the EUs. To deal with the intractability of the problem, we decomposed the MINLP problem into multiple tractable sub-problems using ADMM technique. The simulation results confirmed the effectiveness of the proposed ADMM-based algorithm, and showed the advantage of jointly optimizing the bandwidth allocation for training and inference process over that allocating dedicated bandwidth for AI model transmissions. Nonetheless, new training and inference procedures can be designed to further reduce the communication cost and improve the training accuracy, which we would like to leave for future study.

Appendix A
Proof of Proposition 1

Let μ_j be the non-negative Lagrangian multiplier associated with the constraint $T_{j,N} \geq a_j(y_j + \tau_{j,N}^{\rm md} + C(1-\iota_j)L_j/f_j) - \tau_{j,N-1}^{\rm md}$ in $\mathcal{U}_j$. Then, a partial Lagrangian function of (18) can be given as

$$\mathcal{L}_j = S^I(u_j) + \mu_j\left[a_j\left(y_j + \tau_{j,N}^{\rm md} + C(1-\iota_j)L_j/f_j\right) - \hat{T}_{j,N}\right], \quad (25)$$

where $\hat{T}_{j,N} = T_{j,N} + \tau_{j,N-1}^{\rm md}$. By taking the partial derivative of $\mathcal{L}_j$ with respect to f_j and setting $\frac{\partial \mathcal{L}}{\partial f_j} = 0$, the optimal structure of the local CPU frequency can be given as

$$f_j = \min\left(\sqrt[3]{\mu_j/(w_j^{\rm e} 2k_{\rm c})}, f_{\max}\right). \quad (26)$$

Then, the optimal value of $\hat{T}_{j,N}$, denoted as $\hat{T}_{j,N}^*$, can be discussed in the following two cases:

1) $0 \leq \iota_j < 1$*:* In this case, $a_j = 1$ and $\mu_j > 0$. Otherwise, $f_j = 0 \Rightarrow \mathcal{L}_j \to \infty$, leading to an unbounded result for problem (18). Therefore, according to the KKT condition $\mu_j[(y_j + \tau_{j,N}^{\rm md} + C(1-\iota_j)L_j/f_j) - \hat{T}_{j,N}] = 0$, we have that

$$\hat{T}_{j,N}^* = y_j + \tau_{j,N}^{\rm md} + C(1-\iota_j)L_j/f_j. \quad (27)$$

2) $\iota_j = 1$*:* In this case, $a_j = 0$. According to (9), we have

$$\hat{T}_{j,N}^* = \tau_j^{\rm pr} + CL_j/f_{\rm s}. \quad (28)$$

Thus we complete the proof of Proposition 1.

References

[1] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[2] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the Internet of Vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, Feb. 2020.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, Jan. 2019.

[4] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[5] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[6] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[7] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 63–71.

[8] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[9] M. Chen *et al.* (2019). *A Joint Learning and Communications Framework for Federated Learning Over Wireless Networks.* [Online]. Available: http://arxiv.org/pdf/1909.07972v2

[10] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[11] J. Yan, S. Bi, and Y.-J. A. Zhang, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 235–250, Jan. 2020.

[12] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," in *Proc. ICLR Workshop Track*, vol. 17, May 2016, pp. 1–10.

[13] Z. Hou, R. Chi, and H. Gao, "An overview of dynamic-linearization-based data-driven control and applications," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4076–4090, May 2017.

[14] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.