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

Xián 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

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
that an arbitrary part of the task can be offloaded.

that the inference task data does not contain privacy-sensitive contents, such as that the inference task data does not affect the following inference process. For simplicity, we assume that the inference task data does not contain privacy-sensitive contents, such that the inference task data does not affect the following inference process. For simplicity, we assume 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 computing [4]. 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 \(\tau_{j,n}^{lt}\) and \(\tau_{j,n}^{mu}\) 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 \(\tau_{j,n}^{lt}\) and energy consumption (denoted as \(e_{j,n}^{lt}\)) 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 \(\tau_{N}\). Then, each EU downloads the updated model for a duration denoted by \(\tau_{j,n}^{md}\), 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 \(\tau_{1,N}^{10}\) and \(\tau_{2,N}^{10}\) amount of time, respectively. The first EU starts downloading the model from time instant \(\tau_{N}\) for a duration \(\tau_{1,N}^{md}\), with which it performs local inference with duration \(\tau_{1,N}\). Meanwhile, the ES starts computing the task of EU \(j\) for a duration of \(\tau_{j,N}^{ce}\) once its task data is completely received (e.g., from the time instant \(\tau_{j,n}^{pe}\) 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.

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 computing [4]. 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 \(\tau_{j,n}^{lt}\) and \(\tau_{j,n}^{mu}\) 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 \(\tau_{j,n}^{lt}\) and energy consumption (denoted as \(e_{j,n}^{lt}\)) 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 \(\tau_{N}\). Then, each EU downloads the updated model for a duration denoted by \(\tau_{j,n}^{md}\), and retrains the model in the
0). Since an EU downloads the model only when performing local task processing, we have that \( a_j = 1 \) when \( 0 \leq t_j < 1 \) and \( a_j = 0 \) when \( t_j = 1 \). Then, the total energy consumption of an EU \( j \in M \) in the \( N \)-th iteration can be written as

\[
E_{j,N} = \frac{1}{j_{N}} + e_{jn}^{\text{mu}} + e_{jn}^{\text{to}} + a_j (e_{jn}^{\text{md}} + e_{jn}^{f}),
\]

(8)

Correspondingly, the execution latency is

\[
T_{j,N} = \max \{ a_j (\tau_{N} + \tau_{j,N} + \tau_{j,N} + \tau_{j,N} + \tau_{j,N} + \tau_{j,N} + \tau_{j,N} + \tau_{j,N} + \tau_{j,N} + \tau_{j,N}) \},
\]

(9)

where \( \tau_{jn} = \max \{ \tau_{j,n} \} \) denotes the time that both the AI model and task data are ready for processing the task of EU \( j \). Notice that \( \tau_{jn} \) 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 alleviate the negative impact of the energy and latency cost induced by the communication overhead of federated learning during model training. Specifically, we solve

\[
\min \sum_{j \in M} \left( w_j^T E_{j,N} + w_j^T T_{j,N} \right)
\]

(10a)

s.t. \( \sum_{j \in M} (b_j + b_j) \leq 1, b_j, b_j \geq 0, \forall j \in M \),

(10b)

\( 0 \leq f_j \leq f_{\text{max}}, \forall j \in M \),

(10c)

\( \{ a_j, t_j \} \in D_j, \forall j \in M \),

(10d)

\( \{ \tau_{N}, \tau \} \in T \),

(10e)

where \( b = \{ b_j, b_j \}, a = \{ a_j \}, t = \{ t_j \}, f = \{ f_j \}, \tau = \{ \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn} \} \), \( \{ \tau_{jn} \} \), and \( \tau \in M \). \( w_j^T \) and \( w_j^T \) 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 EU. In (10d) and (10e), \( D_j \) describes the constraints on model download decision and task splitting ratio, and \( T \) captures the feasible region of the time costs \( \{ \tau_{N}, \tau \} \), 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 problem 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., \( \tau_{j,N} f_j \) and \( u_j f_j \)) and the binary variables \( a_j \) make (10) intractable. A potential method to solve (10) is to fix \( \tau \) and \( a \), then the remaining problem is convex and can be solved via off-the-shelf tools (e.g., interior point method). The optimal \( \{ a, t \} \) can be found via enumerating all the \( \prod_{j=1}^{M} (\Delta t_j) \) possible combinations, where \( \Delta t_j \) is a sufficiently small precision interval of \( t_j \). However, when \( M \) becomes large, the exhaustive search on \( \{ a, t \} \) 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 \( x = \{ x_j \}, y = \{ y_j \}, \) and \( z = \{ z_j \} \), where \( j \in M \), and convert (10) into an equivalent form as below:

\[
\begin{align*}
\min_{u_j, \tau_N} & \sum_{j \in M} \left( q_j(u_j) + g(\hat{\tau}_N, b) \right) \\
\text{s.t.} & \quad x_j = b_j, y_j = \tau_N, z_j = b_j, \forall j \in M, \\
& \quad z_j, y_j, \tau_N \geq 0, \forall j \in M, \\
& \quad (10c)-(10e).
\end{align*}
\]

(12a)

where \( u_j = \{ x_j, y_j, z_j, a_j, t_j, f_j, \} \), \( \tau_N \), and \( u \) = \{ \hat{u}_j, \hat{j} \}. \) In the objective function (12a), \( q_j(u_j) = w_j^T E_{j,N} + w_j^T T_{j,N} \) and

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

(13a)

where \( \mathcal{F} = \{ \hat{\tau}_N, b | \sum_{j \in M} (b_j + b_j) \leq 1; b_j \geq 0, b_j \geq 0, \forall j \in M; \hat{\tau}_N \geq 0 \} \). Let \( a = \{ a_j \}, \beta = \{ \beta_j \} \) and \( \xi = \{ \xi_j \}, \forall j \in M \), 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

\[
\mathcal{L}_A(u, \tau, \omega) = \sum_{j \in M} q_j(u_j) + g(v) + \sum_{j \in M} \alpha_j (x_j - b_j)
\]

\[
+ \sum_{j \in M} \beta_j (y_j - \hat{\tau}_N) + \sum_{j \in M} \xi_j (z_j - b_j) + \frac{c}{2} \sum_{j \in M} (x_j - b_j)^2
\]

\[
+ \frac{c}{2} \sum_{j \in M} (y_j - \hat{\tau}_N)^2 + \frac{c}{2} \sum_{j \in M} (\beta_j - b_j)^2,
\]

(14)

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

\[
K(\omega) = \min_{u,v} \mathcal{L}_A(u, \tau, \omega), \quad \text{s.t.} (12c), (10c)-(10e).
\]

(15)

At the higher level, we solve the dual problem

\[
\max_{\omega \geq 0} K(\omega).
\]

(16)

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

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

\[
\begin{align*}
u^I+1 = \arg \min_{u_j \in U_j} S^I(u_j), \quad \text{s.t.} (12c), (10c), (10d).
\end{align*}
\]

(18)

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

\[
\mathcal{T} = \{ T_j, \tau_N, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn}, \tau_{jn} \}, \forall j \in M \}
\]

\[
\mathcal{T}_j = \{ a_j, t_j | a_j \in \{ 0, 1 \}, 0 \leq t_j \leq 1, (1-a_j)(1-t_j) = 0 \} \setminus \{ a_j = 1, t_j = 1 \}
\]

(11)

Authorized licensed use limited to: SHENZHEN UNIVERSITY. Downloaded on March 10,2021 at 01:37:20 UTC from IEEE Xplore. Restrictions apply.
the data-driven-based search [13]). 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 $t_j$, the local CPU frequency of the $j$-th EU at the optimum of (18) can be given as

$$f_j = \min \left( \sqrt{\frac{\mu_j}{(w^0_j + 2e_c)\cdot f_{\text{max}}}} \right),$$

where $\mu_j$ 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 $T_{j,N} = \hat{T}_{j,N} - \tau_{j,N}^0$, where

$$\hat{T}_{j,N} = \begin{cases} y_j + \tau_{md,j,N}^0 + C(1 - t_j)L_j/f_j, & 0 \leq t_j < 1, \\ \frac{c_j}{\rho_j} + CL_j/f_b, & t_j = 1. \end{cases}$$

**Proof:** Please refer to Appendix A.

Proposition 1 implies that, as long as some task data is processed locally (i.e., $t_j < 1$), 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 $u^{t+1}$, we update $v^{t+1}$ by solving the following convex optimization problem:

$$\begin{align*}
\min_{\{\tau_j, b_j\} \in \mathcal{F}} & \sum_{j \in M} \left[ y_j \left( \tau_j^{t+1} - b_j \right) + \beta_j \left( \tau_j^{t+1} - \gamma_j \right) N_j + \zeta_j \left( \tau_j^{t+1} - b_j \right) \right] \\
+ & \frac{c}{2} \sum_{j \in M} \left[ \left( \tau_j^{t+1} - b_j \right)^2 + \left( \gamma_j - \gamma_j^{t} \right)^2 + \left( \zeta_j - \zeta_j^{t} \right)^2 \right].
\end{align*}$$

The optimal solution is in the following closed-form:

$$v_j^{t+1} = \frac{1}{M} \sum_{j' \in M} \left( \tau_{j'}^{t+1} + \gamma_{j'}^{t+1} \right), \quad \gamma_j = \left( \gamma_j^{t+1} - \theta_j^{t+1} \right) / e_j, \quad \zeta_j = \left( \zeta_j^{t+1} - \theta_j^{t+1} \right) / e_j,$$

where $j = 1, \ldots, M$ and $(\cdot)^{t+1} \triangleq \max(\cdot, 0)$. $\theta^*$ 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 $u^{t+1}$ and $v^{t+1}$, we update the multipliers $\omega^t$ using the sub-gradient method as below:

$$\begin{align*}
\alpha_j^{t+1} &= \alpha_j^t + c \left( \tau_j^{t+1} - b_j^{t+1} \right), \quad j = 1, \ldots, M, \\
\beta_j^{t+1} &= \beta_j^t + c \left( y_j^{t+1} - \gamma_j^{t+1} \right), \quad j = 1, \ldots, M, \\
\zeta_j^{t+1} &= \zeta_j^t + c \left( \gamma_j^{t+1} - \tau_j^{t+1} \right), \quad j = 1, \ldots, M.
\end{align*}$$

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

$$\begin{align*}
\sum_{j \in M} \left[ \left| \tau_j^{t+1} - b_j^{t+1} \right| + \left| y_j^{t+1} - \gamma_j^{t+1} \right| + \left| \gamma_j^{t+1} - \tau_j^{t+1} \right| \right] &\leq \epsilon_1, \\
\sum_{j \in M} \left[ \left| \tau_j^{t+1} - b_j^{t+1} \right| + \left| y_j^{t+1} - \gamma_j^{t+1} \right| + \left| \gamma_j^{t+1} - \tau_j^{t+1} \right| \right] &\leq \epsilon_2,
\end{align*}$$

where $\epsilon_1$ and $\epsilon_2$ denote the absolute tolerance and relative tolerance [14], respectively. Otherwise, the $(l + 2)$-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 $M$.

### 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 $h_j = c h_j$. Here, $c$ is an independent exponential random variable of unit mean, which captures the small-scale channel fading effect. $h_j$ denotes the average channel gain that follows a path-loss model $h_j = G_A (\frac{3 \times 10^6}{4.7 f_j d_j})^\sigma$, where $G_A = 4.11$ captures the total antenna gain, $f_c = 2.4$ GHz represents the carrier frequency, and $\sigma$ is the path-loss exponent, respectively. We assume all EUs are identical in task data bulk $L$, maximum transmit power $p_{\text{max}}$, receiver power $p_{\text{md}}$, and weighting factors $w_j$ and $z_j$. Here we drop the subscript $j$ for convenience. Without loss of generality, we set $\tau_{j,N} = \tau_{j,N}^0 = 1$ s and $e_{j,N} = 0.04$ Joule, $\forall j \in \mathcal{M}$.

The other parameters used in simulation are listed in Table I.

<table>
<thead>
<tr>
<th>TABLE I SIMULATION PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_c = 33$ dBm</td>
</tr>
<tr>
<td>$B = 20$ MHz</td>
</tr>
<tr>
<td>$\delta_1^c = 427$ dBm/KHz</td>
</tr>
<tr>
<td>$w_j = 0.2$</td>
</tr>
<tr>
<td>$c_1 = 1.5 M \times 10^{-4}$</td>
</tr>
</tbody>
</table>

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 $e_{j,N} = 0, \forall j \in \mathcal{M}$).
- **Group bandwidth equalization (GBE):** The uplink and downlink transmissions occupy equal bandwidth $\frac{B}{2}$ each.
- **Optimal:** exhaustively enumerating all the $2^M$ combinations of task offloading choices, where $\Delta t_{j,N} = 0.2, \forall j \in \mathcal{M}$.

The ADMM-based method adopts the golden-section search to update the value of $t_j$, 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 $M$. In this simulation, the path-loss exponent is set as $\sigma = 2.9$. We also evaluate the impact of $\Delta t$ on the performance of exhaustive searching by considering $\Delta t = 0.5, 0.2, 0.1$, respectively. The results show that the performance of exhaustive searching
and LCO at a small GBE method, the WSEL performance is better than ECO

\[ \Delta D \]

536 IEEE WIRELESS COMMUNICATIONS LETTERS, VOL. 10, NO. 3, MARCH 2021

\[ \Delta D \]

achieves a similar grade to the ADMM-based method. This is
to leave for future study.

Now, new training and inference procedures can be designed to further reduce the communication transmissions. Nonetheless, new training and inference processes over that allocating dedicated bandwidth for AI model

simulation results confirmed the effectiveness of the proposed 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.

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

\[
\mathcal{L}_j = S^j(u_j) + \mu_j[y_j + \tau_{md,j,N} + C(1 - \tau_j)L_j/f_j - \hat{\tau}_{j,N}],
\]

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

\[
f_j = \min \left( \sqrt{\frac{\mu_j}{(w^2k_\tau)}}, \frac{1}{f_{\text{max}}} \right).
\]

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

1) \( 0 \leq \tau_j < 1 \): In this case, \( a_j = 1 \) and \( \mu_j > 0 \). Otherwise, \( f_j = 0 \Rightarrow \mathcal{L}_j \rightarrow \infty \), leading to an unbounded result for problem (18). Therefore, according to the KKT condition

\[
\mu_j[y_j + \tau_{md,j,N} + C(1 - \tau_j)L_j/f_j - \hat{\tau}_{j,N}] = 0,
\]

we have that

\[
\hat{\tau}_{j,N}^* = y_j + \tau_{md,j,N} + C(1 - \tau_j)L_j/f_j.
\]

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

\[
\hat{\tau}_{j,N}^* = \tau_{md,j}^* + CL_j/f_s.
\]

Thus we complete the proof of Proposition 1.

REFERENCES


