Uncertainty-Aware Multitask Allocation for Parallelized Mobile Edge Learning

Duncan J. Mays, Sara A. Elsayed, Hossam S. Hassanein
School of Computing, Queen's University, Kingston, ON, Canada
duncan.mays@queensu.ca, selsayed@cs.queensu.ca, hossam@cs.queensu.ca

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10436750

***Abstract*—Harvesting the profuse yet underutilized computational resources of IoT devices, also referred to as Extreme Edge Devices (EEDs), can significantly curtail the delay in parallelized Mobile Edge Learning (MEL). However, EEDs are user-owned devices, which causes them to experience a highly dynamic user access behavior. Such dynamicity can lead to uncertainty in the available computation and communication capabilities of learners. In this paper, we propose the Minimum Expected Delay (MED) scheme. MED is the first data allocation scheme in MEL that accounts for uncertainty in learners' capabilities and enables multitask allocation. Given the state probabilities of learners, MED strives to minimize the sum of the maximum expected delay of all tasks, while abiding by certain training time and budget constraints. Towards that end, MED formulates the data allocation problem as an Integer Linear Program (ILP) and makes uncertainty-aware decisions. We conduct rigorous experiments on a real testbed of Jetson Nano devices. Extensive performance evaluations show that MED outperforms a representative of state-of-the-art uncertainty-naive schemes by up to 11%, 11%, 42%, and 5% in terms of training time, satisfaction ratio, data drop rate, and occupancy time, respectively. In addition, MED approaches a baseline scheme that assumes a perfect knowledge of the learners' states, yielding a gap of up to 10%, 5%, and 14% in terms of satisfaction ratio, data drop rate, and occupancy time, respectively.**

***Index Terms*—Parallel Learning, Mobile Edge Learning, Multitask Allocation, Uncertainty, Extreme Edge Devices.**

I. Introduction

With the advent of the Internet of Things (IoT), it is anticipated that 41.6 billion IoT devices will be connected to the Internet by 2025, triggering 79.4 zettabytes of data [1]. Due to the time-sensitive nature of such data, it is expected that 90% of analytics will be conducted at the edge to avoid the excessive delay associated with data transmission to remote data centers in cloud computing [2].

Extreme Edge Computing (EEC) has emerged as a computing paradigm that can drastically curtail the delay. It has been adopted by several industrial entities [3]. EEC helps reduce the delay by exploiting the prolific yet underutilized computational resources of end devices, also referred to as Extreme Edge Devices (EEDs) [4] [5] [6]. Although each EED may have limited resources on its own, their collective power can be significant. Consequently, integrating EEDs with Mobile Edge Learning (MEL) provides a propitious edge learning paradigm for a wide spectrum of applications, such as Tactile Internet, ChatGPT, smart cities, and virtual and augmented reality [7] [8]. Note that MEL enables Machine learning (ML) models to be collaboratively trained on a collection of edge devices.

MEL can be classified into two main categories; Parallelized Learning (PL) and Federated Learning (FL) [7] [8]. In PL, a central orchestrator is responsible for sending data subsets to each learner, while the learners in FL train on their own locally stored data [8]. PL is used when the orchestrator lacks the necessary computational resources to train an effective model, thus prompting the workload to be distributed among several learners [8]. FL, on the other hand, allows learners to use models that are trained on a larger dataset, while keeping their own data private [7]. Both PL and FL involve multiple learners that train independent models in parallel, which are later aggregated by the orchestrator to create a more generalized model [9] [10]. One of the challenging issues encountered in PL is system heterogeneity, where learners possess different levels of computation and communication capabilities.

Failing to consider system heterogeneity can constrain PL to the level of performance achieved by its least capable learner [8]. To resolve this issue, existing schemes offer several solutions. These include varying the size of the learners' local models [11], enabling learners to iterate over their local data a different number of times [8] [9], or assigning different amounts of data to each learner [12].

Data allocation schemes that allocate different amounts of data to each learner can minimize staleness, which helps improve the learning accuracy [8]. However, existing schemes fail to account for uncertainty in learners' capabilities. In particular, since EEDs are user-owned devices, they undergo a dynamic user access behavior. Thus, at any given time during training, users can access their devices to stream a video, play a video game, or run any other intensive application/learning task, which can profoundly affect the computation and communication capabilities of learners [5]. This dynamic behavior and resource contention can trigger a high level of uncertainty in the training capabilities of learners, which can drastically increase the training time and data drop rate [5]. Such uncertainty imposes a challenging data allocation issue that has been mostly overlooked in MEL. In addition, most existing schemes fail to consider scenarios where the orchestrator receives multiple rather than single learning tasks.

In this paper, we propose the Minimum Expected Delay (MED) scheme. MED enables multitask data allocation that accounts for uncertainty in the different states at which the

learners can be, and the probabilistic impact of such states on their capabilities. MED models variations in a given learner's performance as uncertainty in the learner's capabilities, and allocates data for multiple tasks in the presence of such uncertainty. Given the state probabilities of each learner (i.e., learner's capabilities), MED formulates the data allocation problem as an Integer Linear Program (ILP) to minimize the sum of the maximum expected training delay of all incoming tasks, while abiding by certain budget and deadline limits. Our contributions can be summarized as follows:

- We account for uncertainty in learners' capabilities and make uncertainty-aware data allocation decisions in PL. For the first time in the literature, we consider the dynamic user access behavior of EEDs and minimize the training time of PL under uncertainty.
- We foster multitask data allocation that enables multiple tasks to contend for the available learners' resources, while maintaining certain time and budget constraints.
- We conduct extensive experiments on a real testbed of Jetson Nano devices.

We implement MED using a custom-built Python MEL framework, Axon [13]. We evaluate the performance of MED by comparing it to a representative of uncertainty-naive schemes [14] [15], as well as a baseline scheme that adopts the ideal yet unpractical case, where the orchestrator has a perfect knowledge of the learners' states. Extensive evaluations show that MED yields significant improvements of up to 11%, 11%, 42%, and 5% in terms of training time, satisfaction ratio, data drop rate, and occupancy time, respectively, compared to the uncertainty-naive scheme. In addition, MED approaches the ideal scheme, with a gap of up to 10%, 5%, and 14% in terms of satisfaction ratio, data drop rate, and occupancy time, respectively.

The remainder of the paper is organized as follows. In Section II, we highlight some related work. In Section III, we provide a detailed description of the proposed scheme (MED). In Section IV, we discuss the performance evaluation and results. In Section V, we highlight our conclusions and future directions.

II. Related Work

Various studies [8] [12] [16] have investigated the issue of system heterogeneity in MEL. To address this issue, existing schemes typically adjust the workload assigned to learners according to their capabilities [12] [16]. This can be achieved by adopting one of three approaches; altering the number of learning iterations performed by each learner [11], adjusting the size of the model assigned to each learner [12], or varying the amount of data allocated to each learner [8] [17] [18].

In [11], learners perform varying numbers of local updates during each global communication cycle. However, this approach leads to stale updates, which in turn can negatively impact the accuracy of the training process. Staleness, which is a measure of the redundancy of two parameter updates from different learners, is used to determine the extent to which such updates provide redundant information.

HeteroFL [12] addresses the issue of system heterogeneity by distributing models of varying sizes to learners based on their capabilities. To accomplish this, the orchestrator prunes the central model into various sizes using the lottery ticket technique [19]. These sub-models are then transmitted to the learners. After global aggregation, the orchestrator converts all sub-models back to their original sizes before aggregating them. Although effective, this approach can be resource-intensive, since the orchestrator must refine the model into parameter sets of diverse sizes.

In [8] [17] [18], the approach of changing the amount of data allocated to each learner results in significant improvements compared to the aforementioned approaches. In particular, this approach is computationally inexpensive for the orchestrator and is capable of eliminating staleness. In [18], a decentralized data allocation scheme that relies on benchmarking is proposed. The underlying benchmarking scheme endeavors to estimate the rate (in samples per second) at which each learner can train a certain model. However, such schemes assume that the orchestrator has a single rather than multiple incoming learning tasks. In addition, they assume that the computation and communication capabilities of learners remain static throughout the learning task. Thus, they fail to account for uncertainty.

In contrast to existing schemes, we propose an uncertainty-aware data allocation scheme that enables multitask allocation in the presence of uncertainty in learners' capabilities. We also conduct experiments on a real testing environment rather than relying on simulations.

III. Minimum Expected Delay (MED)

In MED, learners are provided with multiple models, one to be trained for each learning task, and the orchestrator determines the amount of data to be allocated to each learner for each task. MED allocates data for each learning task given a pool of learners with uncertain computation and communication capabilities. In this section, we present the underlying system model and problem formulation of MED.

A. System Model and Overview

Consider a set of n learners $W = \{w_1, w_2, ..., w_n\}$, and a set of m requesters $R = \{r_1, r_2, ..., r_m\}$ that subscribe to the service provided by the orchestrator. The orchestrator is a central entity that is responsible for allocating the computing resources of learners to serve incoming requests. Each requester $r_i \in R$ has a parallel learning task that involves training D_i data shards among the learners such that a certain training deadline T_i is not exceeded. Let Ω_i be the model parameters of the learning task of requester r_i. The size (in bytes) of each data shard for each request r_i is denoted α_i.

Each learner $w_j \in W$ is recruited by the orchestrator in exchange for some incentives. In particular, each learner w_j sets a certain unit price p_j per data shard. Note that all requests must be served within a certain budget B. At any given time, each learner w_j can be in one of a set $S = \{s_1, s_2, ..., s_h\}$ of h possible states, each of which corresponds to a potential user

activity, such as idle, training, downloading, etc. Each state induces different computation and communication capabilities for the learner. Note that such states can be estimated by the orchestrator using historical information of the learners [20]. For each learner w_j, each state $s_{j,k} \in S$ triggers different benchmarks $c_{i,j,k}$ and $b_{j,k}$ with probability $\gamma_{j,k}$, where $\gamma_{j,k}$ indicates the level of certainty in the learner's state. This probability can be acquired using uncertainty quantification methods [21]. The benchmarks are determined using the subset benchmarking technique [18]. The benchmark $c_{i,j,k}$ represents the compute power of learner w_j for request r_i when the learner is in state $s_{j,k}$. In other words, it is the rate (in data samples per second) at which w_j can train the model of request r_i when the learner is in state $s_{j,k}$. The benchmark $b_{j,k}$ represents the networking capability (in bytes per second) of learner w_j when the learner is in state $s_{j,k}$. Similar to other works, we assume that the channel is perfectly reciprocal [17].

It is paramount that each learner uploads the trained parameters of each request r_i to the orchestrator by the deadline T_i. The orchestrator aggregates all the uploaded parameters and restarts the training regime in the next Global Update Cycle (GUC). Typically, learners that do not upload their parameters by the deadline are not represented in the aggregated parameters that are trained in the next GUC. This causes their work to be wasted. To improve the performance, we use learner halting. Upon task assignment, the orchestrator informs each learner of the deadline of the task assigned to it. Each learner periodically checks this deadline throughout their local update routine. If a learner exceeds the deadline, it halts training and uploads its partially trained parameters to the orchestrator, regardless of its progress in the training task. These partial updates are then aggregated as normal, with reduced weighting due to being trained on fewer data samples.

B. Problem Formulation

The delay of a parallel process is the delay of the longest sub-process that needs to be completed. In PL, this means that the training time in a GUC is the longest training time rendered by all the learners training for the corresponding task [14] [15]. Thus, in an uncertainty-naive data allocation approach [14] [15], the best way to minimize the training delay is to minimize the maximum delay of learners. However, such an approach fails to account for uncertainty in learners' capabilities, where a learner's network bandwidth and training rate depend on the state of the learner. In MED, we account for uncertainty by considering the expected delay given the learners' state probabilities.

The main objective of MED is to minimize the sum of the maximum expected training delay of all tasks. The expected training delay of each learner w_j when assigned $d_{i,j}$ data shards from task r_i is denoted $\delta_{i,j}$, and is calculated by summing the delay in each state multiplied by the probability of that state, as given by Eq. 1. Note that the training delay of learner w_j at state $s_{j,k}$ is the sum of the time it takes to download the model parameters and the training data, train the model, and upload its parameters. MED strives to achieve its objective while abiding by certain budget and deadline limits.

$$\delta_{i,j} = \sum_{k \in S} \gamma_{j,k} \left[\frac{2\Omega_i + \alpha_i d_{i,j}}{b_{j,k}} + \frac{d_{i,j}}{c_{i,j,k}} \right] \tag{1}$$

We formulate the data allocation problem as an Integer Linear Program (ILP), where the decision variable $d_{i,j}$ indicates the amount of data shards of request r_i that is allocated to learner w_j. The problem formulation is given by Eq. 2.

$$\min_{d_{i,j} \in \mathbb{Z}_{\geq 0}} \sum_{i \in R} \max_{j \in W} \delta_{i,j} \tag{2a}$$

subject to:

$$\delta_{i,j} \leq T_i, \forall r_i \in R, \forall w_j \in W \tag{2b}$$

$$\sum_{w_j \in W} d_{i,j} = D_i, \forall r_i \in R \tag{2c}$$

$$\sum_{r_i \in R} \sum_{w_j \in W} d_{ij} * p_j \leq B \tag{2d}$$

$$\sum_{r_i \in R} d_{i,j} \geq D_l, \forall w_j \in W \tag{2e}$$

$$d_{i,j} > 0 => d_{z,j} = 0, \forall r_z \in R, z \neq i, \forall w_j \in W \tag{2f}$$

Constraint (2b) ensures that the expected training delay of each task r_i does not exceed its deadline T_i. Constraint (2c) ensures that for each task r_i, the total number of data shards assigned to all learners is equal to the corresponding size of the task's training data set D_i. Constraint (2d) ensures that the total recruitment cost of all the learners used to train all tasks does not exceed a certain budget limit B. Constraint (2e) is used to guarantee that each learner is assigned a lower bound of data shards D_l. This is to ensure the utilization of all learners, which can positively affect accuracy. Constraint (2f) ensures that each learner is assigned data shards from at most one request.

IV. Performance Evaluation

In this section, we evaluate the performance of MED compared to the MinMax Training Time-Ideal (MMTT-Ideal) scheme and the MinMax Training Time-Uncertain (MMTT-Uncertain) scheme [14] [15]. Note that MMTT-Ideal adopts the ideal case, where the orchestrator has a perfect knowledge of the learners' capabilities at any given state. Though unrealistic, MMTT-Ideal considers the case where there is no uncertainty, and thus acts as an upper bound on the reachable potential of uncertainty-naive schemes. In contrast, MMTT-Uncertain is a representative of state-of-the-art uncertainty-naive schemes [14] [15]. In MMTT-Uncertain, the orchestrator assumes that the learners' capabilities remain the same during the training process when in fact they do change over time due to the dynamic user access behavior.

We use the following performance metrics: 1) The average training time, which is the average time needed to complete training the learning models of the tasks. 2) The satisfaction

TABLE I: Description of the 3 ML tasks MNIST˙FFN, MNIST˙CNN, and Fashion.

	Architecture	Number of Iterations	Deadline	Number of Data Shards
MNIST_FFN	ThreeNN	2	55 (sec)	60
MNIST_CNN	ConvNet	1	40 (sec)	60
Fashion	FashionNet	1	40 (sec)	20

ratio, which is the ratio of the number of tasks that finish training before their specified deadline to the total number of tasks. 3) The average data drop rate, which is the number of the data samples that have been dropped (i.e., not trained on) in a task, divided by the total number of data samples in the task's dataset, averaged among all tasks. 4) The average occupancy time, which is the amount of time the learners spend working on a task divided by the total time to complete the trial, averaged among all tasks. Note that learners spending more time sitting idle reduces this metric.

A. Experimental Setup

We implement MED, MMTT-Ideal, and MMTT-Uncertain on a real testbed of 10 Jetson Nanos, each with 4 GB memory, and a 912 MHz processor. This implementation is done using our custom-built Python framework Axon [13]. Axon is an open-source, simple, and easy to access MEL framework. We use the Gurobi optimizer [22] to solve the optimization problem.

Experiments are conducted using three training tasks, described in Table I. The architecture of the Neural Network (NN) trained for the three tasks MNIST_FFN, MNIST_CNN, and Fashion is denoted ThreeNN, ConvNet and FasionNet, respectively. Note that ThreeNN consists of three hidden feed-forward layers of size [500, 300, 100]. ConvNet uses two 3×3 convolutional layers with 6 and 16 filters, respectively, and a feed forward layer with 200 units. FashionNet uses two 3×3 convolutional layers with 16 and 32 filters, respectively, and two feed-forward layers with 1000 and 50 units each. All neural networks are trained over 5 GUCs with a learning rate of 0.0001.

We use the subset benchmarking scheme [18] with a benchmark size of 20 shards to determine the learners' computation and communication characteristics. Prior to the experiments, each learner w_j runs network and compute benchmarks for each task r_i and state $s_{j,k}$ to determine $b_{j,k}$ and $c_{i,j,k}$. The number of states and number of tasks are set to 3, the budget is set to 300$, and the minimum number of data shards D_l allocated to each learner is set to 1. We model the variance in the unit price specified by learners using a bounded normal distribution. At the beginning of each trial, learners determine their price by sampling a normal distribution of mean 1.0 and variance 0.5. This sample is then bounded to the range [0.1, 1.5].

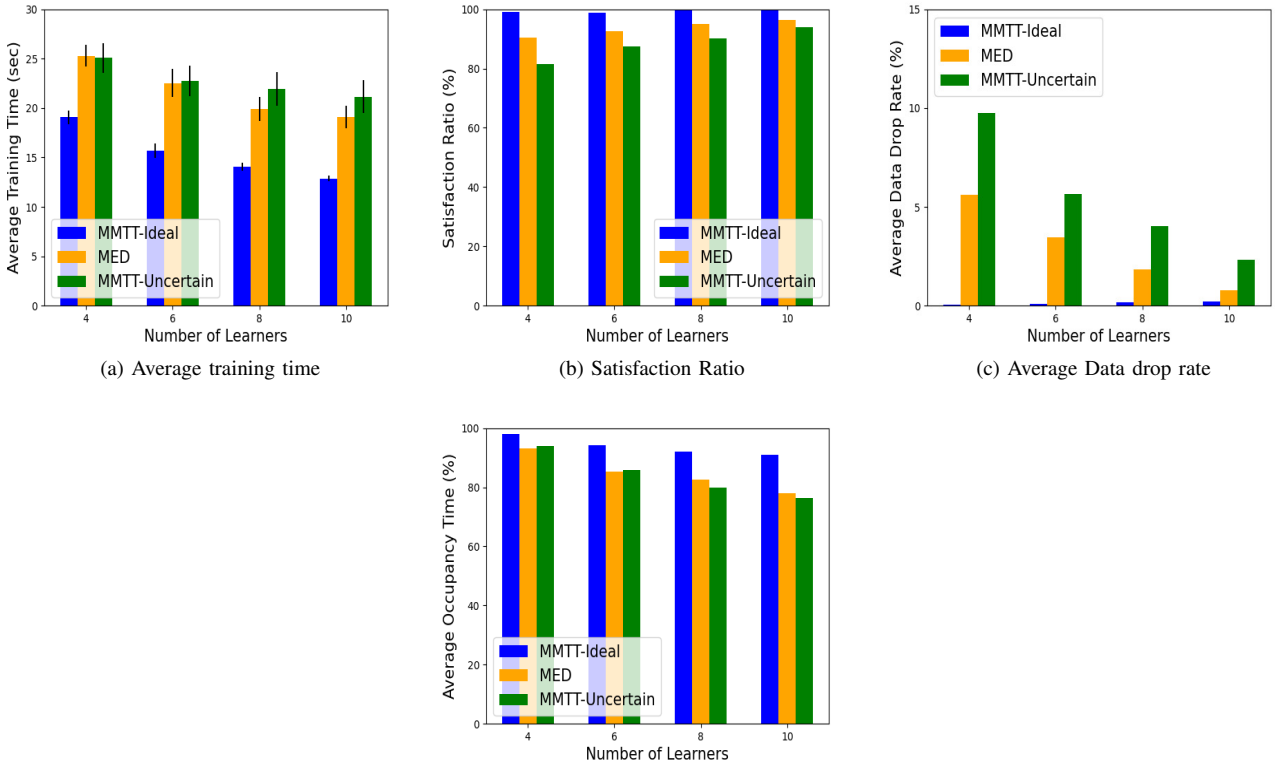
We simulate variations in learners' capabilities using a set of discrete states, one corresponding to each stressor function. The learner's state distribution is set at the beginning of each trial, and then sampled to determine the state at the beginning of each local training routine. We use three states; idle, training and downloading. The idle state does not involve any stressor function and represents the full capabilities of the learner. The training state represents resource contention with another training task, and is represented via the multiplication of two 900×900 matrices during the training loop. The download state represents contention for network bandwidth and is represented by downloading a 900×900 matrix of random data.

We instantiate the learners' state distribution by feeding a random one-hot vector into a heated softmax function. First, a one-hot vector with elements representing each state is created by sampling a uniform distribution across the states. This vector is then fed into a softmax function, which accepts both a vector and a heat parameter, and outputs a probability distribution across the learner states. The softmax heat parameter controls the spread of the output distribution. High heat values correspond to uniform, high entropy state distributions, whereas low heat values correspond to distributions that are strongly skewed to one state and have low entropy. The resulting state distribution is sampled at the beginning of each training routine to set the learner's state. The state heat parameter is set to 0.5.

B. Results and Analysis

In our experiments, we evaluate the performance of MED, MMTT-Ideal, and MMTT-Uncertain over varying number of learners. All experiments are repeated 40 times for each instance, and simulation results are presented at a confidence level=95%. The rendered confidence intervals are shown in Figure 1(a). Since the confidence intervals are negligible, they are not explicitly depicted in the remaining figures for clarity of presentation.

Figure 1(a) depicts the average training time of MED, MMTT-Ideal, and MMTT-Uncertain over varying number of learners. Note that the average training time decreases as the number of learners increases in all schemes. This can be attributed to the fact that as the number of learners increases, the chance of allocating lower amount of data to each learner increases, which increases the learner's speed of training the allocated data, thus reducing the average training time. MMTT-Ideal yields the upper bound on the potential improvement in training time. This is since MMTT-Ideal has perfect knowledge of the learners' states at the beginning of each GUC, and reallocates data to ensure the lowest training time as the learners' states change. MED approaches MMTT-Ideal, with a gap of up to 32%. Note that MED significantly outperforms MMTT-Uncertain, with an improvement of up to 11%. This is because, as opposed to MED, MMTT-Uncertain does not account for uncertainty. Thus, MMTT-Uncertain allocates data based on inaccurate benchmark scores that represent the learners' capabilities when they are in a different



(a) Average training time

(b) Satisfaction Ratio

(c) Average Data drop rate

(d) Average time occupancy

Fig. 1: Performance results for a global update cycle of MED, MMTT-Ideal, and MMTT-Uncertain over varying numbers of learners..

state than the one they are actually in. Thus, there is a high risk of allocating more data to less capable learners, which can prolong the training time in MMTT-Uncertain. In other words, MMTT-Uncertain is much more prone than MED to overestimating the capabilities of resource-poor learners, and underestimating the capabilities of resource-rich learners.

We conduct the same experiment to assess the satisfaction ratio of MED, MMTT-Ideal, and MMTT-Uncertain over varying number of learners. As depicted in Figure 1(b), the satisfaction ratio increases as the number of learners increases in all schemes. This is because the higher the number of learners, the fewer the shards of data that are allocated to each learner. Consequently, learners tend to complete their training routines faster, and are thus more likely to complete the task before the deadline. MMTT-Ideal has near perfect satisfaction ratio for all number of learners. This is due to the fact that it has perfect knowledge of the learners' states at each GUC and can reallocate to maintain efficiency as the learners' states change. MED closely approaches MMTT-Ideal in satisfaction ratio as the number of learners increases, yielding a gap of up to 10%. MED also outperforms MMTT-Uncertain, with an improvement of up to 11%. This can be attributed to the fact that MED has a probabilistic knowledge of the learners' states, whereas MMTT-Uncertain assumes that the states remain fixed throughout the training process, and it does not possess any knowledge about their dynamic change. As a result, MMTT-Uncertain tends to over-allocate data to weak learners while underutilizing strong learners, leading to a lower likelihood of finishing tasks on time than MED.

Figure 1(c) shows the average data drop rate of MED, MMTT-Ideal, and MMTT-Uncertain over varying number of learners. The drop rate is a metric that quantifies the number of data batches that are allocated but not trained on. Thus, learners halting early in their training routine increases the drop rate. Note that as the number of learners increases, the data drop rate decreases. This is because as the number of learners increases, fewer data shards need to be assigned to each learner. Thus, learners are more likely to complete the whole task within the specified time. MMTT-Ideal has an average data drop rate of nearly 0% for all numbers of learners. This is due to the fact that it can reallocate data in between GUCs with full and perfect knowledge of the learners' states, which helps ensure that the whole task is completed. MED closely approaches MMTT-Ideal as the number of learners increases, yielding a small gap of up to 5%. It can also be observed that MED significantly outperforms MMTT-uncertain by up to 42%. This is since MMTT-Uncertain is unaware of the uncertainties in the learners' training capabilities, and is thus more likely to allocate data to learners that have a higher risk of not completing training before the deadline.

Figure 1(d) depicts the average occupancy time of MED, MMTT-Ideal, and MMTT-Uncertain over varying number of learners. Note that the average occupancy time declines for all schemes as the number of learners increases. This is because as the number of learners increases, it is more likely that at least one learner uploads their update late, thus forcing the others to sit idle for some time, which leads to wasting resources. Note that MMTT-Ideal represents the upper bound on the reachable potential improvement in occupancy time. This is due to its perfect knowledge of the learners' states, which reduces the training time, thus reducing the risk of learners sitting idle waiting for others to finish training on their allocated data shards. MED yields the second best average occupancy time, with a gap of up to 14% with MMTT-Ideal. MED outperforms MMTT-Uncertain by up to 5%. This can be attributed to the fact that MED accounts for uncertainty in learners' states, which helps reduce the average training time compared to MMTT-Uncertain, thus reducing the risk of learners sitting idle waiting for other learners to finish.

V. Conclusion

In this paper, we have proposed the Minimum Expected Delay (MED) scheme. MED fosters data allocation for multiple learning tasks in the presence of uncertainty in learners' capabilities in Parallelized Learning (PL) at the extreme edge. Given the state probabilities of learners's capabilities, MED makes uncertainty-aware data allocation decisions that strive to minimize the sum of the maximum expected delay of all tasks, while maintaining certain budget and deadline limits. Extensive evaluations on a real testing environment have shown that MED outperforms a representative of uncertainty-naive schemes by up to 11%, 11%, 42%, and 5% in terms of training time, satisfaction ratio, drop data rate, and occupancy time, respectively. Evaluations have also shown that MED approaches the ideal scheme, with a gap of up to 10% and 5% in terms of satisfaction ratio and drop data rate, respectively. In the future, we plan on using data replication schemes to account for the probabilistic risk of learners failing to finish training on time.

Acknowledgment

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number ALLRP 549919-20, and a grant from Distributive, Ltd.

References

[1] "The Growth in Connected IoT Devices is Expected to Generate 79.4ZB of Data in 2025", Business Wire, [Online], Available: https://www.businesswire.com/news/home/20190618005012/en/The-Growth-in-Connected-IoT-Devices-is-Expected-to-Generate-79.4ZB-of-Data-in-2025-According-to-a-New-IDC-Forecast, 2019.

[2] K. Gyarmathy, "Iot statistics and trends to know in 2022", [Online], Available: https://www.vxchnge.com/blog/iot-statistics (Accessed 2023, April 10).

[3] "Distributive Ltd.", [Online], Available: https://kingsds.network/ (Accessed 2023, April 10).

[4] R. F. El Khatib, S. A. Elsayed, N. Zorba and H. S. Hassanein, "Optimal Proactive Resource Allocation at the Extreme Edge," *IEEE International Conference on Communications (ICC)*, Seoul, Korea, pp. 5657-5662, 2022.

[5] "M. De'bas, S. A. Elsayed and H. S. Hassanein, "Multitiered Worker-Oriented Resource Allocation at the Extreme Edge," *IEEE Global Communications Conference (GLOBECOM)*, Rio de Janeiro, Brazil, pp. 5674-5679, 2022.

[6] "HomeEdge", [Online], Available: https://wiki.lfedge.org/display/HOME/Home+Edge+Project (Accessed 2023, April 10).

[7] W. Y. B. Lim et al., "Federated Learning in Mobile Edge Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031-2063, thirdquarter 2020.

[8] U. Mohammad and S. Sorour, "Adaptive Task Allocation for Mobile Edge Learning," *IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*, Marrakech, Morocco, pp. 1-6, 2019.

[9] H. Yu, S. Yang, and S. Zhu, "Parallel Restarted SGD with Faster Convergence and Less Communication: Demystifying Why Model Averaging Works for Deep Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5693–5700, 2019.

[10] K. A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. M. Kiddon, J. Kone˘cn´y, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards Federated Learning at Scale: System Design", arXiv preprint arXiv:1902.01046, 2019.

[11] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When Edge Meets Learning : Adaptive Control for Resource-Constrained Distributed Machine Learning", IEEE Conference on Computer Communications (INFOCOM), Honolulu, HI, USA, pp. 63-71, 2018.

[12] E. Diao, J. Ding, V. Tarokh, "HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients" [Online] Available: https://arxiv.org/abs/2010.01264, 2021.

[13] "Axon", [Online], Available: https://github.com/DuncanMays/axon-ECRG (Accessed 2023, April 10).

[14] B. Yuan, Y. He, J. Q. Davis, T. Zhang, T. Dao, B. Chen, P. Liang, C. Re, and C. Zhang, "Decentralized Training of Foundation Models in Heterogeneous Environments," arXiv preprint arXiv:2206.01288, 2022.

[15] H. Qi, E. R. Sparks, and A. S. Talwalkar, "Paleo: A Performance Model for Deep Neural Networks," *International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[16] Z. Wei, S. Gupta, X. Lian, and J. Liu, "Staleness-Aware Async-SGD for Distributed Deep Learning," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2350–2356, 2016.

[17] U. Mohammad, S. Sorour, and M. Hefeida, "Task Allocation for Mobile Federated and Offloaded Learning with Energy and Delay Constraints", *IEEE International Conference on Communications Workshops (ICC Workshops)*, Dublin, Ireland, pp. 1-6, 2020.

[18] D. J. Mays, S. A. Elsayed and H. S. Hassanein, "Decentralized Data Allocation via Local Benchmarking for Parallelized Mobile Edge Learning," *International Wireless Communications and Mobile Computing (IWCMC)*, Dubrovnik, Croatia, pp. 500-505, 2022.

[19] J. Frankle, M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks", *7th International Conference on Learning Representations (ICLR)*, New Orleans, Louisiana, 2019.

[20] R. Kain, S. A. Elsayed, Y. Chen, and H. S. Hassanein, "Multi-step Prediction of Worker Resource Usage at the Extreme Edge", *in Proceedings of the 25th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pp. 25–32, 2022.

[21] A. Amini, W. Schwarting, A. Soleimany, and D. Rus, "Deep Evidential Regression", Advances in Neural Information Processing Systems, vol. 33, pp. 14927–14937, 2020.

[22] Gurobi, "Gurobi Optimizer Reference Manual", [Online], Available: http://www.gurobi.com (Accessed 2023, April 10).