Edge-MSL: Split Learning on the Mobile Edge via Multi-Armed Bandits

Taejin Kim
Carnegie Mellon University
tkim2@andrew.cmu.edu

Jinhang Zuo
Caltech & UMass Amherst
jinhangzuo@gmail.com

Xiaoxi Zhang
Sun Yat-sen University
zhangxx89@mail.sysu.edu.cn

Carlee Joe-Wong
Carnegie Mellon University
cjoewong@andrew.cmu.edu

***Abstract*—The emergence of 5G technology and edge computing enables the collaborative use of data by mobile users for scalable training of machine learning models. Privacy concerns and communication constraints, however, can prohibit users from offloading their data to a single server for training. Split learning, in which models are split between end users and a central server, somewhat resolves these concerns but requires exchanging information between users and the server in each local training iteration. Thus, splitting models between end users and geographically close edge servers can significantly reduce communication latency and training time. In this setting, users must decide to which edge servers they should offload part of their model to minimize the training latency, a decision that is further complicated by the presence of multiple, mobile users competing for resources. We present Edge-MSL, a novel formulation of the mobile split learning problem as a contextual multi-armed bandits framework. To counter scalability challenges with a centralized Edge-MSL solution, we introduce a distributed solution that minimizes competition between users for edge resources, reducing regret by at least two times compared to a greedy baseline. The distributed Edge-MSL approach improves trained model convergence with a 15% increase in test accuracy.**

I. Introduction

The increasing prevalence of Internet-of-Things (IoT) devices that can collect and analyze data has fueled the rise of many new applications and services, e.g., in smart homes and cities [1]. To facilitate these new applications, many recent works have proposed various distributed and federated machine learning methods that allow models to be trained on data that is physically distributed across multiple IoT devices. These methods generally require user devices to train local models on their own data, with occasional synchronization at a central coordinator, e.g., a cloud server. Thus, they provide better privacy guarantees with less communication than sending all data to cloud servers for model training [2].

Vanilla federated learning architectures place a considerable computing burden on users, by requiring them to compute multiple iterations of gradient descent on a regular basis. For large models, such computational burdens can be prohibitive, and many federated learning algorithms attempt to reduce this burden [2]. In particular, recent works have developed the idea of **federated split learning**, in which part of a model resides on user devices and part resides at the central coordinator [3]–[5]. Typically, the model is a neural network, and each user maintains a set of individual network layers, while all users share another set of layers maintained by the central coordinator as seen in Figure 1. As in vanilla federated learning, training proceeds in rounds, during each of which a client will compute multiple gradients of its local model, communicating with the shared model at the coordinator as needed [6]. By locating part of the model at the coordinator, split learning thus reduces the computing done at the local device. At the same time, user privacy is not compromised, as only pre-processed data is sent to the coordinator [7].

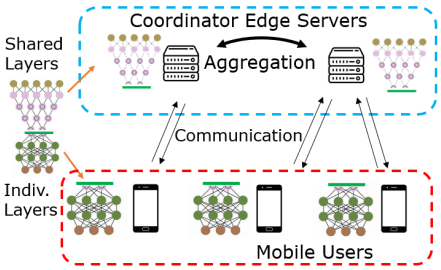


Fig. 1: Split learning framework where neural network layers are split between users and edge servers.

Deploying split learning in practice often relies on the emerging paradigm of *edge computing*, in which devices may access computing resources on nearby edge servers [8]. Many cellular operators are moving towards providing MEC (multi-access edge computing) [9] services near or co-located with 5G base stations. These edge servers' geographic proximity reduces their communication latency to end users, which benefits split learning frameworks, especially for latency-sensitive applications like smart health, as they require multiple exchanges of intermediate data and gradients between the different parts of the model during training [3]. However, 5G users are often within range of multiple suitable base stations [10], each of which can maintain copies of the shared layers in anticipation of expected user demand [11], [12]. Thus, users face a fundamental research question: **to which edge servers should users offload their shared layers?** To the best of our knowledge, *our work is the first to answer this question in the presence of user mobility*, e.g., for autonomous vehicles, which changes users' optimal offloading decisions over time.

A. Challenges: Split Learning in Mobile Edge Computing

Mobile split learning introduces new challenges beyond those of previous mobile edge computing optimization [13].

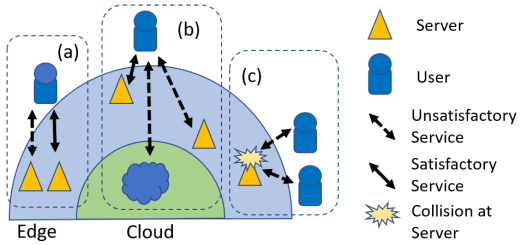


Fig. 2: The successful participation of a client in split learning depends on the (a) innate capability of edge server to complete training task in time, (b) distance between user and server, and (c) coordination amongst different offloading users.

Figure 2 illustrates that users must select edge servers accounting for (a) the compute and communication capability of the server to complete the training task on time, (b) their current physical distance to the server (and thus communication latency), and (c) collisions with other users, i.e., whether the edge server has the capacity to serve all users that offload to it. These characteristics moreover change as users move, resulting in changing edge server preferences and user competition.

Users' selections of edge servers are made difficult by the inherent **uncertainty** in their received services. Not only can access latency depend on rapidly varying and location-dependent wireless channel conditions [14], but users generally do not know how much competition from other users they will face at a given edge server. Thus, prior works have proposed **centralized, learning-based allocation methods** for assigning users to edge servers [13], [15], which can easily coordinate user competition. Indeed, similar combinatorial multi-armed bandit frameworks have been proposed to select which federated learning clients can advance model training the most in any given training round, which can be used to prioritize users competing for edge server resources [16], [17].

Such centralized learning-based methods, however, often struggle to account for **dynamics in users' optimal choices of edge servers**. These naturally arise due to *user mobility*, which causes users' edge server preferences to change over time as they move closer to or further from different edge servers. Tracking these dynamics in a centralized manner may be difficult. Thus, these centralized methods generally *scale poorly to large numbers of users and servers* [13]. Furthermore, the central server may require privacy-sensitive information from individual users in centralized algorithms, in order to determine the optimal allocation [16]. Thus, a promising alternative is a *distributed algorithm* that **allows individual users to select their own edge servers** [18].

Distributed algorithms also have the advantage of allowing users to choose between edge services offered by different providers who may not coordinate with each other. However, without carefully designed coordination mechanisms, they cannot prevent *collisions*, i.e., multiple users offloading to the same edge server and overwhelming its limited resources, leading to poor experience for all offloading users (case (c) of Figure 2). This coordination should also adapt to user mobility; however, since other users' movements are unknown to an individual user, from an individual user's perspective the distributed scenario induces a *non-stationary* environment, which introduces new learning challenges [19]. These are exacerbated in the split learning setting, since users' differing local datasets and individual model layers can make some users more "useful" to the training than others, complicating the optimal allocation of users to edge servers as these more useful users should be prioritized [3]. To the best of our knowledge, *our work is the first to propose such a decentralized coordination algorithm under end user mobility.*

B. Edge-MSL: Mobile Split Learning in Edge Computing

In this paper, we propose Edge-MSL (Mobile Split Learning on the Edge), a contextual and combinatorial bandits formulation aiming to solve the problem of deciding which users should offload their training of split models to which edge servers. In more detail, our **technical contributions** are as follows:

- We design the first system that optimizes edge computing allocations for split learning using a *contextual and combinatorial multi-player bandits framework* to jointly address split learning performance at different edge servers, collisions between user requests, and the relation between user mobility and latency. We show that combinatorial bandit algorithms continue to achieve sublinear regret in this setting (Theorem 1).
- We next consider *distributed user offloading algorithms* and are the first to show that, due to user collisions and mobility, they necessarily have linear (or worse) regret (Theorem 2). We quantify the effect of user collisions on this worst-case regret (Theorem 3), which motivates our design of a novel reservation-based algorithm that reduces collisions by using edge servers to mediate them. We analytically quantify our algorithm's regret relative to naïve distributed algorithms. To the best of our knowledge, this work is the first to extend multi-player bandits with collisions to non-stationary settings.
- We perform *extensive evaluations* of our distributed solutions on real mobility traces [20] for split learning. Our distributed solution, Edge-MSL:D, incurs less regret than other methods that do not consider collisions and mobility by at least a factor of 2. Models trained via Edge-MSL:D converge faster, to a test accuracy 10% to 25% higher for the CIFAR-10 and 15% for the CIFAR-100 data set.

The remainder of this paper is organized as follows. Section II contrasts Edge-MSL with related works. Section III presents the system model and the fact that the Edge-MSL model allows us to formulate a linear integer optimization problem for edge allocations. Sections IV and V present our centralized and distributed Edge-MSL solution methods, respectively, and theoretically examine their performance. Finally, we *experimentally validate our work* compared to prior approaches in Section VI. We conclude in Section VII. All proofs are shortened to sketches to conserve space.

θ_u^P, θ^H	Personalized (for user u) and shared layers of split learning model	r_u^t	Reward received by user u at time t. Sum of rewards across all users is $r(I_t, Y, X_t, \mu)$
Y_u	Data quality constant $\in [0,1]$ for user u. Y is the vector of Y_u $\forall u$.	$\mu_{u,s}$	Parameter for unscaled reward distribution of user u offloading to server s. μ is the vector of $\mu_{u,s}$ $\forall u, s$.
$p^u_{l_1,l_2}$	Probability user u moves from location l_1 to l_2 at each round, following the probability transition matrix P_u	$X_{l_u^t,s}$	Context variable $\in \{0,1\}$ to scale received reward for latency based on location of user u and server s
$i^t_{u,s}$	Binary indicator that returns 1 if user u offloads to server s at time t and 0 otherwise. I_t is the vector of $i^t_{u,s}$ $\forall u, s$	j^t_s	Binary indicator variable that returns 1 if a collision occurs at server s at time t and 0 otherwise.

TABLE I: Variables used for the Edge-MSL formulation.

II. Related Work

In **federated learning**, edge servers can be used as learning coordinators [21], to run training updates themselves [22], or to store shared layers of a federated split learning model [8]. Variants of **split learning** aim to improve its communication latency or reduce its needed computing resources [3]–[5] when users offload to a given server. However, none of these works consider how split learning end users should choose the edge server to which they offload their training.

The efficient **allocation of users to edge servers** for generic computing jobs [13] has been extensively studied under the framework of multi-armed bandits. Prior works have proposed bandit solutions for a single user, e.g., [23] examines a non-stationary environment without mobility, and [24], [25] consider energy constraints. Other works [25], [26] consider the decentralized multi-player bandits framework, while [27] introduces a sophisticated collision mechanism that splits rewards amongst competing users and [28] considers the related problem of decentralized users learning to associate with heterogeneous wireless base stations. However, these works do not consider user mobility. Other work proposes to handle mobility with contextual bandits, with edge servers acting as mediators for user actions [18], but does not consider user collisions. Bandit frameworks are also used in the context of (mobile) vehicles acting as edge servers [29]–[31].

Algorithmically, our work falls into the research line of **multi-player multi-armed bandits** [32] and **contextual multi-armed bandits** [33]. Prior work solves the combinatorial bandits problem when different players have different expected rewards for arms, which change following a Markovian pattern [34] or are stationary [35]. Other works have considered multi-player bandits with heterogeneous user rewards [36]. Unlike Edge-MSL, these formulations do not include contextual variables, as is needed to model user heterogeneity in split learning. Prior work on (single-player) contextual combinatorial bandits learns the relationship between context variables and reward, while in our setting the context is a simple scaling factor, allowing better performance bounds [37]. Other work on contextual multi-player bandits for wireless channel allocation assumes i.i.d. (independent and identically distributed) user mobility [38], which removes the non-stationarity challenge. Prior work on single-player non-stationary switching bandits has shown that they may achieve linear regret relative to the optimal policy [19], [39], [40]. However, these works do not consider multi-player bandits, in which the non-stationary environment is induced by other users, as in our split learning framework.

III. System Model and Problem Formulation

Physical Model. We consider an edge computing service provider with S servers to service $U \leq S$ users, each with a split learning task. A "server" can represent a share of a virtual container within a physical server; we abstract these resources as servers for simplicity. For example, if a single VM has the capacity to support five users, we abstract it as five "servers". Note that some "servers" may also be located in the resource-rich cloud, allowing us to assume $U \leq S$. We let $[N]$ represent the set $\{1, 2, ..., N\}$. We consider discrete time steps $t \in [T]$, which also function as rounds in the bandits and split learning setting. While our model is agnostic to the exact time step length, in practice, the time steps would likely last a few minutes. At this granularity, we can meaningfully represent user movement around a city via their locations at different times as well as the duration of a single training round of split learning. Following prior work [41], [42], all users enter the system with a learning task to offload at time $t = 1$ and exit at $t = T$. Table I summarizes our notation.

Split Learning Framework. Every user u performs split learning following Algorithm 1, training a neural network with personalized layers (θ_u^P) at the user's local device, and shared layers (θ^H) at edge servers. The personalized layers serve as the input layers, ensuring raw data remains on the user's local device. Only intermediate data between the personalized and shared layers, as well as labels needed for computing loss for back propagation, are sent to the edge server from the user. At the end of each training round, the shared layers are aggregated. Each round has a given wall-clock time deadline; to prevent stragglers, users who do not finish their model updates by the deadline discard this update [4].

User Mobility. We consider $l \in [L]$ discretized locations from which users will offload processes to edge servers, each with a distance of $d_{l,s}$ to server $s \in [S]$. User u's location for time step t is denoted as l_u^t. The movement of users from one location to another is depicted as a Markov chain as in [41], where a transition probability matrix P_u captures the mobility pattern for user u, and $p^u_{l_1,l_2}$ is the transition probability for user u from location l_1 to l_2 at the next time step.

Service Offloading and Reward. At each time step t, if a user u offloads their service (shared layers) to server s, the indicator variable $i^t_{u,s} = 1$, and $i^t_{u,s} = 0$ if otherwise. The reward a user u receives from offloading to server s is $r^t_{u,s} \sim i^t_{u,s} Y_u X_{l_u^t,s}$Bernoulli$(\mu_{u,s})$. The quality of a user's data, Y_u, is a known constant representing the expected per round contribution of the data mapped within the range $Y_u \in [0,1]$. For example, previous works define Y_u values

Algorithm 1: Mobile split learning framework.

Input: Users $u \in [U]$, edge servers $s \in [S]$, user data $(x_u, y_u) \sim D_u$, model personalized (input) layers $\theta^P_{c,t}$, and shared (output) layers θ^H_t.

Initialization: Randomly initialize $\theta^P_{c,t}$ and θ^H_t.

For each round $t = 1, ..., T$

1) *Server Selection*: Each client $u \in [U]$ selects edge server $s \in [S]$ to offload $\theta^H_{c,t}$. Shared layers are downloaded or migrated to chosen edge servers.
2) *Local Training*: Each client u performs at first a forward pass through $\theta^P_{c,t}$ using data at device (x_u), and sends smashed data as well as labels y_u to the edge device holding $\theta^H_{c,t}$. Back propagation begins at the edge server towards the local user.
3) *Aggregation*: All shared layers that perform updates within a deadline ($U^* = [U]_{\tau_u < \tau_G}$) are aggregated across edge servers, i.e., $\theta^H_{t+1} = \sum_{u \in U^*} \frac{1}{|D_u|} \theta^H_{t,u}$.

as a function of the number of data points at a client, as well as the sum of losses for each data point, which quantify the impact of this user's participation on federated learning convergence [16]. Similar expressions can be used in our split learning framework, though the convergence of split learning models is much harder to analytically quantify [6].

The distance-based context variable $X_{l^t_u,s} \in \{0, 1\}$ indicates whether server s is in range of user u based on whether the distance $d_{l^t_u,s}$ exceeds the maximum distance of tolerable latency $d_{\max}$. $X_{l^t_u,s}$ may also be taken as a continuous variable $\in [0, 1]$ if the range depends on other stochastic factors. The expected reward parameter $\mu_{u,s} \in [0, 1]$ represents the probability that a task offloaded by user u to server s is completed within the training round's deadline, separate from the distance-induced latency, e.g., due to straggler compute times [4] or wireless interference.

Collisions. When more than one user offloads to a single edge server, a *collision* occurs between those tasks. We use $j^t_s = 1$ to indicate that a collision occurs at server s and time t (i.e., $\sum_u i^t_{u,s} > 1$); otherwise, $j^t_s = 0$. Each user that collides will have a probability π, which is the same for all users, of being granted access to the server. For example, we may have $\pi = 0$, i.e., no user receives resources from this edge server, or π may be the reciprocal of the number of colliding users, i.e., one user is chosen uniformly at random to receive the edge resources. We focus on these two cases in the rest of the paper; the bulk of our theoretical analysis applies to both.

Optimization Problem Formulation. We define the collection of allocation variables $i^t_{u,s}$ across users and servers at time t as $I_t = (i^t_{u,s})_{u,s}$, the collection of data quality variables as $Y = (Y_u)_u$, the collection of contextual variables as $X_t = (X_{l^t_u,s})_{u,s}$, and the collection of expected reward parameters as $\mu = (\mu_{u,s})_{u,s}$. When $\pi = 0$, the term $r(I_t, Y, X_t, \mu) = \sum_{u,s} i^t_{u,s} Y_u X_{l^t_u,s} \mu_{u,s} (1 - j^t_s)$ denotes the expected reward received across all users given the aforementioned variables taking collisions into account. We show that finding the optimal (i.e., reward-maximizing) global allocation from a set of users to servers can be formulated as a maximum weight matching problem. Thus, we wish to solve:

$$\max_I \sum_{t \in [T]} r(I_t, Y, X_t, \mu) \quad (1)$$
$$\text{s.t.} \sum_{u \in [U]} i^t_{u,s} = 1 \forall s \in [S], \sum_{s \in [S]} i^t_{u,s} = 1 \forall u \in [U]$$

The objective is to find the optimal I_t at every time step that returns the maximum amount of rewards, while the constraints indicate that the optimal solution avoids all collisions, and that users can only offload to a single server every round.

IV. Centralized Mobile Split Learning

We next consider a **centralized** controller that allocates each user to an edge server at every round. This centralized solution prevents reward loss due to collisions, but it requires the controller to communicate with each user every round, including communication of potentially private local data. The remainder of this section elaborates and analyzes our solution algorithm, Edge-MSL:C, a contextual-combinatorial multi-armed bandits approach that achieves sublinear regret.

Edge-MSL:C Algorithm. The expected reward parameters $\mu_{u,s}$ are initially unknown to the central controller and must be learned. Accordingly, we quantify the learning and offloading performance by its regret $\mathcal{R}_\mathcal{C}(T)$, defined as the difference between accumulated rewards received and that of the optimal allocation over all time steps. We define an "arm" as a user-server pair and accordingly denote I^C_t as the chosen arm allocation and I^*_t as the *dynamic optimal solution*.

$$\mathcal{R}_C(T) = \sum_{t \in [T]} r(I^*_t, Y, X_t, \mu) - \sum_{t \in [T]} r(I^C_t, Y, X_t, \mu) \quad (2)$$

The dynamic optimal solution I^*_t may change with time t since we assume the X_t context variables are known to the central controller before the allocation is chosen in each round.

To minimize the regret incurred throughout T time steps, Edge-MSL:C uses an UCB (upper confidence bound)-based approach to effectively learn the reward parameters $\mu_{u,s}$. The central controller obtains user location information l^t_u, estimated expected reward parameters $\bar{\mu}_{u,s}$, the number of previous allocations $z_{u,s}$, and uses the Hungarian algorithm to solve $i^t_{u,s}$ for an allocation that maximizes for round $t \in [T]$:

$$\sum_{u,s} i^t_{u,s} Y_u X_{l^t_u,s} \left(\bar{\mu}_{u,s} + \sqrt{\frac{2ln(t)}{z_{u,s}}} \right). \quad (3)$$

Regret Analysis. We introduce the following two properties of $r(I, X, \mu)$ to bound the regret of Edge-MSL:C. We disregard the data quality term Y as it can be absorbed by μ without altering the following properties and theorems.

Property 1 (Monotonicity)**.** *For any action I, any context X, and any two vectors μ and μ', we have $r(I, X, \mu) \leq r(I, X, \mu')$, if $\mu_{u,s} \leq \mu'_{u,s}$ for all u, s.*

Property 2 (Bounded Smoothness)**.** *For any action I, any context X and any two vectors μ and μ', we have* $|r(I, X, \mu) \leq r(I, X, \mu')| \leq \sum_{u,s} |\mu_{u,s} - \mu'_{u,s}|$.

Both properties are satisfied, providing the problem-independent regret bound for the central UCB approach.

Theorem 1. *(Edge-MSL:C Regret Upper Bound) The regret of Edge-MSL:C is bounded by*

$$\mathcal{R}_C(T) \leq O(U\sqrt{ST \ln T})$$

Proof: We generally follow the proof of Theorem 4 in [43]. We define the reward gap $\Delta^t_{I_t} = r(I^*_t, X_t, \mu) - r(I^C_t, X_t, \mu)$. Given that $\hat{\mu}_t$ represents the UCB value across all arms at time t (i.e., $\hat{\mu}_{u,s,t} = \bar{\mu}_{u,s} + \sqrt{\frac{2ln(t)}{z_{u,s}}}$), the key step is to show the following inequalities.

$$\begin{aligned} r(I^C_t, X_t, \hat{\mu}_t) &\geq r(I^*_t, X_t, \hat{\mu}_t) \\ &\geq r(I^*_t, X_t, \mu) = r(I^C_t, X_t, \mu) + \Delta^t_{I_t}. \end{aligned}$$

Then by Property 2, we have

$$\Delta^t_{I_t} \leq |r(I^C_t, X_t, \hat{\mu}_t) - r(I^C_t, X_t, \mu)| \leq \sum_{u,s} |\hat{\mu}_{u,s,t} - \mu_{u,s}|.$$

All requirements on bounding $\Delta_{S(t)}$ in Lemma 5 from [43] are also satisfied by the algorithm. Hence, we can follow the remaining proofs to derive the desired regret bounds. ∎

While Edge-MSL:C achieves sub-linear regret, the overhead of finding the allocation I^C_t scales poorly with the number of users U and servers S in the system:

Proposition 1. *(Complexity of Edge-MSL:C) The computation overhead of each round of Edge-MSL:C grows at most $O(S^3)$, as the Hungarian algorithm must be run every round [44].*

For split learning in particular, hundreds of users may train a model, which may make this high overhead prohibitive. This overhead, in addition to the need for the central controller to collect user data information Y, motivates a distributed approach, Edge-MSL:D, that maintains lower overhead and privacy-awareness. We discuss Edge-MSL:D next.

V. Distributed Mobile Split Learning

A **distributed** allocation method can reduce the runtime and communication overhead compared to a centralized approach. In this setting, users may easily choose between edge servers owned by different providers without a central coordinator.

In a distributed method, server selection must be done locally by users, as seen in Figure 3. Each user u is assumed to know its own location information l^t_u and have access to its historical service information (i.e., its past received rewards and scaling variables $Y_u, X_{l^t_u,s}$, as well as the presence of collisions). However, it does not have any visibility into other users' received rewards or scaling variables. Similarly, we assume edge servers do not monitor user mobility, to reduce communication and coordination overhead. The remainder of this section elaborates and analyzes distributed bandit approaches, including Edge-MSL:D, that aims to minimize

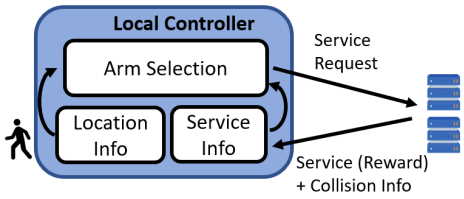


Fig. 3: Flowchart of the distributed approach of solving the allocation problem from users to servers.

collisions between users. For the following analysis, we disregard the data quality term Y as it can be absorbed by μ.

A. Performance Limits of a Distributed Approach

In the distributed case, each user individually aims to learn the expected reward parameters $\mu_{u,s}$. Users must minimize collisions while selecting arms with high reward. We quantify the regret from the distributedly found allocation I^D_t with regard to the optimal allocation I^*_t that solves Equation (1):

$$\mathcal{R}_D(T) = \sum_{t\in[T]} r(I^*_t, Y, X_t, \mu) - r(I^D_t, Y, X_t, \mu) \quad (4)$$

We examine the theoretical characteristics of distributed solutions. In a system that begins with the optimal allocation already found ($I^D_t = I^*_t$), we consider a scenario where a single user moves with probability p_m at every round to cause the optimal solution to change $\left(I^D_t \neq I^*_{t+1}\right)$. As the optimal solution changes, the moving user may offload to (i) a new server, potentially inducing a collision with another user at that server who does not know of the user's movement and thus cannot avoid the collision, or (ii) the same server to avoid collisions despite incurring regret $\left(r(X_{t+1}, I^*_{t+1}) > r(X_{t+1}, I^*_t)\right)$. We denote the resulting expectation (over X_t) of the minimum regret incurred in subsequent rounds due to the user's movement as Δ_{min}; $\Delta_{\text{min}} > 0$, since we consider a finite number of user locations and servers. For example, if $U = S$, the moving user will collide with high probability as another user will be allocated to every other server, unless that other user also switches its server. When $U < S$, as S increases, the probability of collision for the moving user will decrease as more servers will be un-allocated, decreasing Δ_{min}. Theorem 2 shows that any distributed solution has at least linear regret.

Theorem 2 (Distributed Solution Lower Bound)**.** *Assume that a single user has a probability of movement p_m in each time step that alters the contextual variable X_t and the optimal allocation from user to server (i.e., $I^*_t \neq I^*_{t-1}$). Given a distributed algorithm that can discover the new optimal allocation I^*_t with probability of p_o at every round, any distributed solution of problem* (1) *that does not have global knowledge of X_t has expected per round regret lower-bounded:*

$$\lim_{t\to\infty} \frac{\mathbb{E}\left[\mathcal{R}_D(t)\right]}{t} \geq \frac{\Delta_{min}}{1 + p_o/p_m} > 0. \quad (5)$$

Proof: Let at $t = 0$ the optimal allocation $I^*_{t=0}$ be found for $X_{t=0}$. With probability p_m at every time step, a user movement shifts X_t such that the optimal allocation is

altered and at least regret Δ_{min} is incurred in expectation. Thus, an ergodic Markov chain can be used to represent the system, with states π_{opt} and π_{sub} respectively representing the states of optimal and sub-optimal allocation. The state π_{opt} has a transition probability of $p_{\text{opt,sub}} = p_m$, and the state π_{sub} has a transition probability of $p_{\text{sub, opt}} = p_o$. The stationary probability of state $\pi_{\text{sub}} = 1/(1 + p_o/p_m)$. Thus, any distributed solution will incur at least regret Δ_{min} for $1/(1 + p_o/p_m)$ proportion of the rounds in expectation. ■

Since we assume all rewards are bounded, one can also prove a $O(T)$ upper bound on the regret of any allocation algorithm; thus, *any distributed algorithm will have $O(T)$ regret*. While this result is disappointing from a bandits perspective, our split learning framework may still succeed: split learning benefits from having more participating users, but it can successfully train models when a subset of users participate in each training round [16]. Thus, we aim to reduce the constant factor in the regret by considering the two sources of regret: user collisions and pulling sub-optimal arms.

We examine the impact of collisions by comparing two different allocation algorithms. The **greedy distributed** algorithm has each user greedily pull its highest UCB arm at each round while disregarding collisions. The **stationary distributed** algorithm finds a single non-changing allocation I_t^{stat} that is optimal with respect to the expectation of contextual variables $\mathbb{E}[X_t]$, which is constant across time steps. For Markovian user movement, $\mathbb{E}[X_t]$ can be obtained from the stationary distribution of the user mobility Markov chain. The stationary algorithm incurs regret as it does not consider the changing optimal allocation I_t^* as X_t changes, but avoids collisions, as done by existing distributed algorithms [35]. For Theorem 3, we use $r_{\text{opt}} = r(I_t^*, X_t, \mu)$ to represent the optimal expected reward of every round, and $r_{\text{stat}} = r(I_t^{\text{stat}}, X_t, \mu)$ to represent the reward for the stationary algorithm.

Theorem 3 (Collision Rate and Regret)**.** *Assume for a system where reward parameters μ drawn from a uniform distribution are well learned that a static allocation I_t^{stat} has no collisions and in expectation over X_t receives at least $p \geq 1/e$ of the reward of the optimal allocation that takes into consideration dynamic X_t (i.e., $\mathbb{E}[r_{stat}] \geq (1/e) r_{opt}$). Further assume that the number of servers is equal to the number of users ($S = U$) and contextual parameters for each user-server pair is equal in expectation across multiple rounds (i.e., $\mathbb{E}[X_{u,s}^t] = C \; \forall (u, s, t)$). Then for U sufficiently large, the greedy distributed algorithm will always incur equal or higher regret to the stationary policy when collisions lead to no reward for all users.*

Proof: Users of the stationary algorithm will receive the expected reward: $\mathbb{E}[r_{\text{stat}}] = r_{\text{opt}} p \geq r_{\text{opt}}(1/e)$. The incurred regret by the stationary algorithm is then depicted as: $\mathbb{E}[R_{\text{stat}}] = r_{\text{opt}} - \mathbb{E}[r_{\text{stat}}] \leq r_{\text{opt}} - r_{\text{opt}}(1/e) = r_{\text{opt}}(1 - 1/e)$.

Users of the greedy approach will offload to each server with $1/S$ probability, as each server is equally likely to have the highest expected scaled reward. Thus, the probability of a greedy user colliding with any other user each round becomes:

$$P(\text{Coll}_{\text{greedy}}) = 1 - \left(\frac{S-1}{S}\right)^{(U-1)} \tag{6}$$

As the number of servers and users increase the expected collision probability approaches $\lim_{S\to\infty} P(\text{Coll}_{\text{greedy}}) = (1 - 1/e)$. Given that the total reward accrued by the greedy policy is higher than the reward of the optimal solution when ignoring collisions ($\sum_{u,s} X_{l_u,s} \mu_{u,s} i_{u,s}^{t,\text{greedy}} \geq r_{\text{opt}}$), the expected regret of the greedy approach is lower bounded by:

$$\mathbb{E}[\mathcal{R}_{\text{greedy}}] = \sum_{u,s} X_{l_u,s} \mu_{u,s} i_{u,s}^{t,\text{greedy}} (1 - 1/e) \geq r_{\text{opt}}(1 - 1/e).$$

Thus, $\mathbb{E}[\mathcal{R}_{\text{greedy}}] \geq \mathbb{E}[\mathcal{R}_{\text{stat}}]$ when $\mathbb{E}[r_{\text{stat}}] \geq r_{\text{opt}}(1/e)$. ■

Theorem 3 highlights the *collision-production trade-off*, where "production" denotes the user's effective, time-dependent reward. Here, an algorithm that receives a r_{opt}/e reward outperforms the collision prone greedy algorithm.

B. Edge-MSL:D – Reservation Based Distributed Bandits

Algorithm Intuition. Theorem 3 shows that if not controlled, collisions can push a distributed algorithm's regret to be higher than that of a stationary solution that ignores variation in X_t. Thus, a "good" distributed algorithm should limit the number of collisions, which is the main challenge solved by existing distributed bandit work [26], [27], [35]. These works typically utilize a fixed period of time to learn the collision dynamics and then fix on the optimal collision-free allocation by having users "reserve" certain arms.

User mobility, however, reduces the effectiveness of these reservations. For example, users more than d_{max} away from their current locations will not be able to access the reserved servers. Yet even fairly small changes, like a single user departing from the system, require existing distributed bandit algorithms to run a new sequence of discovery rounds to re-learn an optimal allocation [35]. We thus borrow an idea from the non-stationary bandit literature, to adjust the reservation over time as users' reward distributions change [19]. The key question then becomes *how to enforce these reservations and determine how fast they should change*. We can do so with the edge servers, which can inform users of pending reservations.

Edge-MSL:D Description. In **Edge-MSL:D** (Algorithm 2), users aim to offload at every round to the server with the maximum contextual UCB index as calculated in Equation (3). However, when a collision occurs between two or more users, the server with the collision compares the users' estimated production ($X_{l_u^t,s} \bar{\mu}_{u,s}$, which is included in users' service requests) and designates a user with the highest production as "reserving" the server while notifying other colliding users to avoid the server for the same number of rounds (step 3.4). The colliding users with lower production may offload to a server they know is reserved if their expected production increases due to mobility (step 3.2.b). We thus *encourage users with higher $\bar{\mu}_{u,s}$*, who are more likely to successfully complete an update and have more "useful" data as measured by Y_u, to

Algorithm 2: Edge-MSL:D – Distributed collision based reservation UCB method

Input: Users $[U]$, servers $[S]$, reward parameters $\mu_{u,s}$, time horizon T, reservation time $v_{u,s} = 1/p_m$.

Initialization: For all users, set wait time for each server $w_{u,s} = 0, \forall s \in [S]$, production threshold $h_{u,s} = 0, \forall s \in [S]$, reservation time $r_u = 0$, and reservation arm id $a_u = 0$.

For each time step $t = 1, ..., T$

1) **User Movement:** Update user location l_u^t.
2) **Arm selection phase:** For each user $u \in [U]$
   a) If reservation time $r_u > 0$, the user selects the arm it is currently reserving $a_u \in [S]$
   b) If reservation time $r_u = 0$, the user selects the arm with the highest contextual UCB index similar to Equation (3) that has wait time $w_{u,s} = 0$. An arm with $w_{u,s} > 0$ is selected instead if the expected production exceeds that of the reserved user ($X_{l_u^t,s}\bar{\mu}_{u,s} > h_{u,s}$).
3) **Reward phase:** Users pull the arms selected and collect reward information to update $\bar{\mu}_{u,s}$ estimates.
4) **Collision-Communication phase:** When multiple users collide at server s, and one user u_1 has higher estimated production than all other users u_2 (i.e., $X_{l_{u_1}^t,s}\bar{\mu}_{u_1,s} > X_{l_{u_2}^t,s}\bar{\mu}_{u_2,s}$):
   a) If $a_{u_1} = 0$ (user u_1 has not reserved a server) the highest estimated production user u_1 will set $a_{u_1} \leftarrow s$, and set wait time $r_{u_1} \leftarrow v_{u_1,s}$
   b) All other users set arm wait time $w_{u_2,s} \leftarrow r_{u_1}$ and production threshold $h_{u_2,s} \leftarrow X_{l_{u_1}^t,s}\bar{\mu}_{u_1,s}$
5) Update reservation time values for next round
   a) $r_u \leftarrow \min(r_u - 1, 0), \forall u \in [U]$
   b) $w_{u,s} \leftarrow \min(w_{u,s} - 1, 0), \forall (u,s) \in ([U], [S])$
   c) if $r_u = 0$, then $a_u \leftarrow 0$
   d) if $w_{u,s} = 0$, then $h_{u,s} \leftarrow 0$

contribute to the split learning. When comparing the overhead of Edge-MSL:D to that of Edge-MSL:C (Proposition 1), we observe that Edge-MSL:D has a much better runtime:

Proposition 2. *(Complexity of Edge-MSL:D) The computation time of running a single round of Edge-MSL:D grows at $O(S)$, as at every round the maximum UCB value across S servers must be found by each user independently.*

In practice, even when the system size increases, the number of edge servers within the distance $d_{\max}$ from the user likely does not increase. Thus, Edge-MSL:D achieves a sub-linear growth in overhead with respect to S as many servers with $X_{l,s} = 0$ will not be considered for offloading. In order to observe the production-collision trade-off of Edge-MSL:D in comparison to the greedy and stationary distributed algorithms, we estimate Edge-MSL:D's collision rate in Remark 1.

Remark 1 (Edge-MSL:D Collision Rate)**.** *Assume all users in the system have probability of movement from one location to another p_m, and there is probability p_b that a moving user has a high enough production value to interrupt the reserving user via collision (step 3.2.b in Algorithm 2). Under Theorem 3's assumptions, the collision probability per user $u \in [U]$ for each round using Algorithm 2 is estimated by:*

$$P(Coll_{rsv}) = \min\left((1 - 1/e)(p_m + 1 - (1 - p_m p_b)^{U-1}), 1\right) \tag{7}$$

From the perspective of a single user, collisions occur when (i) the user has not reserved a server and collides with another user to then reserve a server, or (ii) another user interrupts a reserved user's service. The expected collision rate of any number of unreserved users at a single server approaches $1 - 1/e$ (Theorem 3). Once the user in question reserves a server for $v_{u,s} = 1/p_m$ rounds, collisions will occur p_m proportion of rounds. While a user is reserved, all other users have a probability of p_m of moving their location and p_b of interrupting the reserved user through collision. Thus, the probability of collision from all other users in $[U]$ interrupting a reservation per round is $1-(1-p_m p_b)^{(U-1)}$. The predicted collision rate of Edge-MSL:D is empirically corroborated in the evaluation of Figure 5 of Section VI-C.

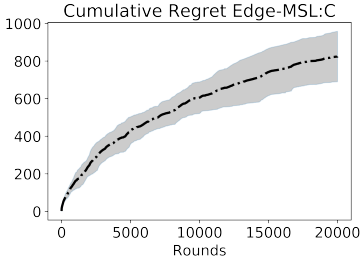
VI. Evaluation

In this section, we numerically evaluate Edge-MSL:C and Edge-MSL:D, validating, and going beyond Section IV's and Section V's results. We aim to show that we have solved the primary research challenges introduced in Section I: designing an effective and low overhead allocation method that (i) takes into consideration user mobility and (ii) reduces collisions to enhance service. After describing our experimental setup, we evaluate the regret and collisions induced by Edge-MSL:C and Edge-MSL:D, and finally the test accuracy and convergence of split learning under these allocation algorithms.

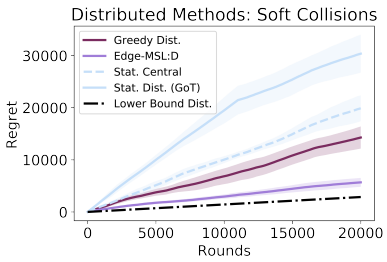
A. Numerical Experiment Setup

We use synthetic server locations spread out uniformly at random within a 10×10 mile area, based on Seoul, an urban area typical of edge computing [45]. The expected Bernoulli reward parameter $\mu_{u,s}$ between every (u,s) user-server pair is drawn uniformly at random between 0 and 1. For simplicity, the data quality parameter Y_u is mapped between $[0,1]$ based on the number of data points each client has; each client trains models on both the CIFAR-10 and CIFAR-100 datasets with 200 to 1800 data points per client. The data is split in a non-i.i.d. manner across clients following Appendix E.1 in [46]. We provide more details on the split learning setup in Section VI-D. Users can be in one of 9 locations that are equally spread throughout the space. Markovian user movements are estimated from the Yonsei/Lifemap mobility dataset taken from Seoul [20]. Time steps (and update deadlines) are five minutes. Unless otherwise stated, $d_{\max} = 4.5$ miles.

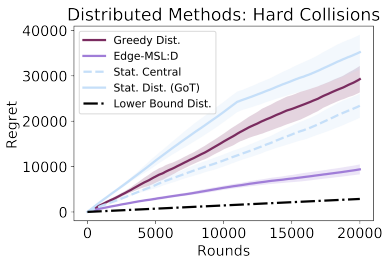
We compare the proposed centralized **Edge-MSL:C** method and distributed **Edge-MSL:D** method, which both consider the context variables X_t, to three baselines. The **greedy**



(a) Cumulative regret for Edge-MSL:C, which achieves sublinear regret.



(b) Cumulative regret of distributed allocation methods for soft-collisions.



(c) Cumulative regret of distributed allocation methods for hard-collisions.

Fig. 4: Our proposed distributed reservation method, Edge-MSL:D, achieves 100+% lower regret and fewer collisions than any method that assumes a stationary contextual variable. The centralized approach, Edge-MSL:C, has the lowest regret.

distributed baseline has each user greedily pull its highest UCB arm at each round while disregarding collisions. The **distributed stationary** baseline follows the Game of Thrones (GoT) approach [35] to find a static allocation from user to servers between all rounds by using a fixed estimate of the contextual variable, through an initial "exploration" phase. The **centralized stationary** baseline similarly attempts to find a static allocation. Unless otherwise stated, all experiments are averaged across 5 trials and run with 9 users and 9 servers. For Section VI-B's regret analysis, we compare a *soft-collision* model, where a single user is chosen randomly by a server when a collision occurs, with a *hard-collision* model, where no colliding users complete their tasks.

B. Comparing Different Allocation Methods

Incurred Regret. We first quantify the regret of the centralized approach in Figure 4a as well as the regret of different distributed approaches for soft collisions in Figure 4b and hard collisions in Figure 4c. Consistent with Theorems 1 and 2, Edge-MSL:C yields a sublinear and a much lower regret than the distributed methods' linear regrets.

Under soft collisions in Figure 4b, the stationary methods perform relatively poorer to the greedy method, as greedy users receive an increased amount of total reward despite collisions. Under hard collisions in Figure 4c, the *centralized stationary method* that does not consider dynamic context variables performs relatively well compared to the *distributed greedy method* as collisions are avoided. However, not considering the contextual variable makes it incur higher regret than both Edge-MSL:D and Edge-MSL:C. The *distributed stationary method* aims to similarly find a static allocation but incurs both a high amount of regret and collisions during the exploration and collision sensing phase of its algorithm. The results indicate that the methods above will lead to few users participating in the split learning training tasks.

The regret lower bound of distributed methods (Equation (5) in Theorem 2) is shown in Figures 4b and 4c. We use the probability of mobility $p_m = 0.02$, estimated from the Yonsei/Lifemap mobility dataset; the probability for an algorithm to correct the allocation per round is conservatively estimated at $p_o = 0.05$; and Δ_{min} is taken as the smallest incurred regret across all experiments with a single collision. Edge-MSL:D comes the closest to this estimated lower bound.

Num. Users	3	5	10	15	20	40
Edge-MSL:D	0.011	0.010	0.011	0.013	0.015	0.022
Edge-MSL:C	0.76	0.91	1.6	2.9	4.8	17

TABLE II: Per round run time in miliseconds experienced by each user for different amount of users and servers.

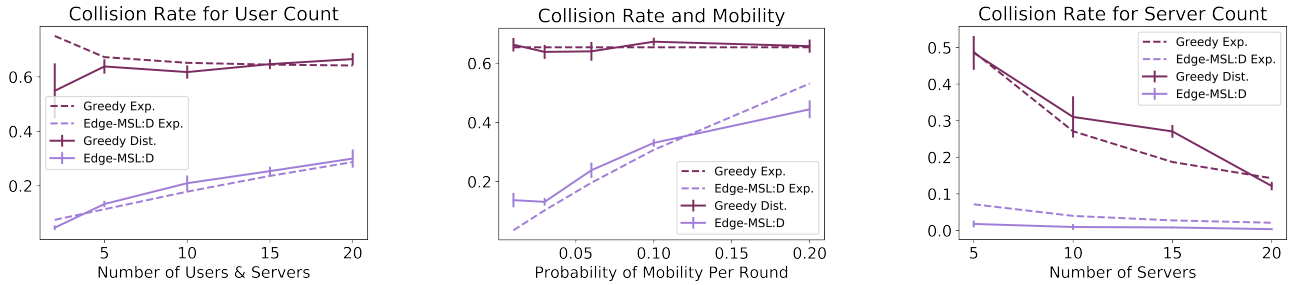
Overhead Analysis. Edge-Alloc:C requires orders of magnitude more time (Table II). The increase in runtime for each method of Edge-Alloc follows Propositions 1 and 2 with respect to the number of users and servers.

C. Collision Prediction Analysis

Consistent with Remark 1, Edge-MSL:D performs worse as user competition and user mobility increase. The empirical collision rates for Edge-MSL:D and the greedy distributed method follow closely the theoretical estimates of Theorem 3 and Remark 1 across different system settings (Figure 5), despite the realistic mobility patterns and its impact on contextual variables X_t. Figure 5a shows that even as the number of users and servers grows under trace-based mobility, the greedy algorithm approaches a $1 - 1/e \approx 0.63$ collision rate, as estimated by Theorem 3. Thus, *even soft collisions would prohibit about one-third of the users from participating* in the split learning training, even before other failures, e.g., due to slow edge server computations [16].

Mobility Patterns. Figure 5b shows the collision rates of different algorithms when user mobility patterns are Markovian, with a probability p_m of moving away from its current location equally to any other location. As p_m increases, the regret incurred by Edge-MSL:D increases: collisions occur more often due to shorter reservation times, and users that do reserve a server have a higher chance from moving away from the server to incur higher regret. The greedy method maintains a consistent, and higher, number of collisions.

Resource Availability. The distributed greedy method has lower collision rates as more servers are available (Figure 5c). Edge-MSL:D successfully maintains low collision rates for various configurations of system resource availability.

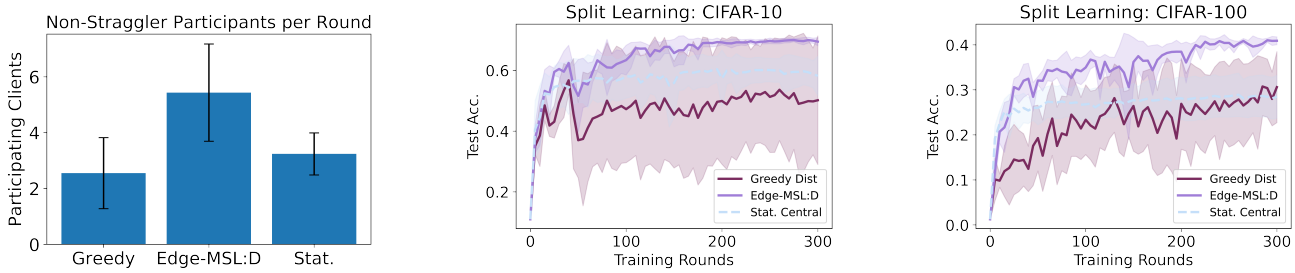


(a) Collision rate as the number of users (equal to servers) increases.

(b) Collision rate for different probability of movement per round p_m.

(c) Collision rate for increasing number servers for a fixed number of users.

Fig. 5: Empirical results follow theoretical results closely for collision rates compared for distributed methods. Context variable X has been altered to a continuous variable in $[0, 1]$ based on distance to allow for more arm options per user.



(a) Average number of non-stragglers per round for the CIFAR-10 split learning.

(b) Test accuracy across rounds for the CIFAR-10 data set .

(c) Test accuracy across rounds for the CIFAR-100 data set.

Fig. 6: Edge-MSL:D achieves higher user participation and prioritization for high data quality users to ensure faster convergence of trained models and higher test accuracy compared to distributed baselines.

D. Split Learning Performance

We finally evaluate split learning tasks on the CIFAR-10 and CIFAR-100 data sets. Here, 20 users perform training by offloading to 20 servers. The models are trained on the MobileNetV2 architecture. Approximately 10% of the model is placed on the user device as personalized layers (θ_c^P), and the rest is placed on the edge server as shared layers (θ^H) so as to minimize the workload of local devices. A hard-collision model is used to simulate a heavily resource-constrained volatile distributed learning scenario [16].

As seen in Figure 6a, Edge-MSL:D successfully increases user participation rate by minimizing collisions as well as considering mobility compared to the baselines. The stationary baseline yields a greater number of participating users than the greedy algorithm, likely because of the greedy algorithm's severe number of collisions. The increased participation rate and the prioritization of higher data quality users leads to higher test accuracy for Edge-MSL:D users, as seen by a 25% increase in Figure 6b for CIFAR-10 and a 15% increase in Figure 6c for CIFAR-100 compared to baseline algorithms.

Increasing the probability of client participation during the training phase in a federated learning setting leads to faster convergence as seen in Theorem 1 of [16]. For a related split learning scenario, Edge-MSL:D empirically demonstrates a higher participation probability of clients, as seen in Figure 6a, leading to faster model convergence relative to the greedy and stationary baselines, as seen in Figure 6b.

VII. Conclusion

While distributed machine learning on the edge has gained traction in recent years, limited resources on the edge as well as dynamic user mobility presents challenges. Often, user competition for resources renders prior methods ineffective. We formulate a linear integer programming problem to quantify the performance of different allocations from users to servers, and utilize a contextual-combinatorial multi-armed bandits framework to solve the problem, whilst respecting resource constraints and user mobility. The centralized Edge-MSL:C algorithm achieves sublinear regret with progressing rounds. We show that all distributed solutions incur at least linear regret due to a lack of coordination between users, and propose the Edge-MSL:D algorithm that reduces regret by a constant factor of at least 2 compared to other distributed baselines while greatly reducing overhead compared to Edge-MSL:C. Edge-MSL:D successfully enhances convergence and test accuracies of models trained by split learning.

Acknowledgements

This work was partially supported by the NSF grant CNS-2103024, NSFC 62102460, Guangzhou Science and Technology Plan Project (202201011392), Guangdong Basic and Applied Basic Research Foundation (2023A1515012982), and Young Outstanding Award under the Zhujiang Talent Plan of Guangdong Province.

REFERENCES

[1] Q. Duan, S. Hu, R. Deng, and Z. Lu, "Combined federated and split learning in edge computing for ubiquitous intelligence in internet of things: State-of-the-art and future directions," *Sensors*, vol. 22, no. 16, 2022.

[2] K. Pfeiffer, M. Rapp, R. Khalili, and J. Henkel, "Federated learning for computationally-constrained heterogeneous devices: A survey," *ACM Computing Surveys*, 2023.

[3] D.-J. Han, H. I. Bhatti, J. Lee, and J. Moon, "Accelerating federated learning with split learning on locally generated losses," in *ICML 2021 workshop on federated learning for user privacy and data confidentiality. ICML Board*, 2021.

[4] E. Samikwa, A. Di Maio, and T. Braun, "Ares: Adaptive resource-aware split learning for internet of things," *Computer Networks*, vol. 218, p. 109380, 2022.

[5] S. Oh, J. Park, P. Vepakomma, S. Baek, R. Raskar, M. Bennis, and S.-L. Kim, "Locfedmix-sl: Localize, federate, and mix for improved scalability, convergence, and latency in split learning," in *Proceedings of the ACM Web Conference 2022*, pp. 3347–3357, 2022.

[6] Y. Li and X. Lyu, "Convergence analysis of split learning on non-iid data," *arXiv preprint arXiv:2302.01633*, 2023.

[7] V. Turina, Z. Zhang, F. Esposito, and I. Matta, "Federated or split? a performance and privacy analysis of hybrid split and federated learning architectures," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pp. 250–260, IEEE, 2021.

[8] Q. Duan, S. Hu, R. Deng, and Z. Lu, "Combined federated and split learning in edge computing for ubiquitous intelligence in internet of things: State-of-the-art and future directions," *Sensors*, vol. 22, no. 16, p. 5983, 2022.

[9] ETSI GS MEC 003 V2.1.1, "Mobile Edge Computing (MEC); Framework and Reference Architecture," 2019. https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf.

[10] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5g systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[11] A. Aral and T. Ovatman, "A decentralized replica placement algorithm for edge computing," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 516–529, 2018.

[12] M. Breitbach, D. Schäfer, J. Edinger, and C. Becker, "Context-aware data and task placement in edge computing environments," in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, pp. 1–10, IEEE, 2019.

[13] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, "A survey and taxonomy on task offloading for edge-cloud computing," *IEEE Access*, vol. 8, pp. 186080–186101, 2020.

[14] W. Gao, Z. Zhao, Z. Yu, G. Min, M. Yang, and W. Huang, "Edge-computing-based channel allocation for deadline-driven iot networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6693–6702, 2020.

[15] B. Sonkoly, J. Czentye, M. Szalay, B. Németh, and L. Toka, "Survey on placement methods in the edge and beyond," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2590–2629, 2021.

[16] F. Shi, W. Lin, L. Fan, X. Lai, and X. Wang, "Efficient client selection based on contextual combinatorial multi-arm bandits," *IEEE Transactions on Wireless Communications*, 2023.

[17] T. Kim, S. Bae, J.-w. Lee, and S. Yun, "Accurate and fast federated learning via combinatorial multi-armed bandits," *arXiv preprint arXiv:2012.03270*, 2020.

[18] O. Tao, X. Chen, Z. Zhou, L. Li, and X. Tan, "Adaptive user-managed service placement for mobile edge computing via contextual multi-armed bandit learning," *IEEE Transactions on Mobile Computing*, 2021.

[19] X. Zhou, Y. Xiong, N. Chen, and X. Gao, "Regime switching bandits," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4542–4554, 2021.

[20] Y. Chon, E. Talipov, H. Shin, and H. Cha, "Crawdad dataset yonsei/lifemap (v. 2012-01-03)," Jan 2012.

[21] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

[22] S. Wang, Y. Ruan, Y. Tu, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-aware optimization of distributed learning for fog computing," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2019–2032, 2021.

[23] A. U. Rahman, G. Ghatak, and A. De Domenico, "An online algorithm for computation offloading in non-stationary environments," *IEEE Communications Letters*, vol. 24, no. 10, pp. 2167–2171, 2020.

[24] S. Ghoorchian and S. Maghsudi, "Multi-armed bandit for edge computing in dynamic networks with uncertainty," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2020.

[25] B. Wu, T. Chen, W. Ni, and X. Wang, "Multi-agent multi-armed bandit learning for online management of edge-assisted computing," *IEEE Transactions on Communications*, vol. 69, no. 12, pp. 8188–8199, 2021.

[26] X. Wang, J. Ye, and J. C. S. Lui, "Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach," 2021.

[27] X. Wang, H. Xie, and J. C. S. Lui, "Multi-player multi-armed bandits with finite shareable resources arms: Learning algorithms &; applications," 2022.

[28] H. Tibrewal, S. Patchala, M. K. Hanawal, and S. J. Darak, "Distributed learning and optimal assignment in multiplayer heterogeneous networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1693–1701, IEEE, 2019.

[29] P. Dai, Z. Hang, K. Liu, X. Wu, H. Xing, Z. Yu, and V. C. S. Lee, "Multi-armed bandit learning for computation-intensive services in mec-empowered vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7821–7834, 2020.

[30] S. Xu, C. Guo, R. Q. Hu, and Y. Qian, "Cooperative multi-player multi-armed bandit: Computation offloading in a vehicular cloud network," in *ICC 2021-IEEE International Conference on Communications*, pp. 1–6, IEEE, 2021.

[31] B. Cho and Y. Xiao, "Learning-based decentralized offloading decision making in an adversarial environment," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 11, pp. 11308–11323, 2021.

[32] K. Liu and Q. Zhao, "Distributed learning in multi-armed bandit with multiple players," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5667–5681, nov 2010.

[33] L. Zhou, "A survey on contextual multi-armed bandits," 2015.

[34] Y. Gai, B. Krishnamachari, and M. Liu, "On the combinatorial multi-armed bandit problem with markovian rewards," 2010.

[35] I. Bistritz and A. Leshem, "Distributed multi-player bandits - a game of thrones approach," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[36] A. Magesh and V. V. Veeravalli, "Decentralized heterogeneous multi-player multi-armed bandits with non-zero rewards on collisions," *IEEE Transactions on Information Theory*, vol. 68, no. 4, pp. 2622–2634, 2021.

[37] L. Chen, J. Xu, and Z. Lu, "Contextual combinatorial multi-armed bandits with volatile arms and submodular reward," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[38] W. Wang, A. Leshem, D. Niyato, and Z. Han, "Decentralized learning for channel allocation in iot networks over unlicensed bandwidth as a contextual multi-player multi-armed bandit game," *IEEE Transactions on Wireless Communications*, vol. 21, no. 5, pp. 3162–3178, 2021.

[39] Y. Chen, C.-W. Lee, H. Luo, and C.-Y. Wei, "A new algorithm for non-stationary contextual bandits: Efficient, optimal and parameter-free," in *Conference on Learning Theory*, pp. 696–726, PMLR, 2019.

[40] A. G. Manegueu, A. Carpentier, and Y. Yu, "Generalized non-stationary bandits," *arXiv preprint arXiv:2102.00725*, 2021.

[41] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2016.

[42] T. Kim, S. D. Sathyanarayana, S. Chen, Y. Im, X. Zhang, S. Ha, and C. Joe-Wong, "Modems: Optimizing edge computing migrations for user mobility," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pp. 1159–1168, 2022.

[43] Q. Wang and W. Chen, "Improving regret bounds for combinatorial semi-bandits with probabilistically triggered arms and its applications," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[44] H. W. Kuhn, *The Hungarian Method for the Assignment Problem*, pp. 29–47. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[45] X. e. a. Liang, "Unraveling the origin of exponential law in intra-urban human mobility," vol. 3, no. 2983, 18 Oct. 2013.

[46] T. Kim, S. Singh, N. Madaan, and C. Joe-Wong, "Characterizing internal evasion attacks in federated learning," 2023.