Edge Computing in the Dark: Leveraging Contextual-Combinatorial Bandit and Coded Computing

Chien-Sheng Yang, *Graduate Student Member, IEEE*, Ramtin Pedarsani, *Member, IEEE*, and A. Salman Avestimehr, *Fellow, IEEE*

***Abstract*— With recent advancements in edge computing capabilities, there has been a significant increase in utilizing the edge cloud for event-driven and time-sensitive computations. However, large-scale edge computing networks can suffer substantially from unpredictable and unreliable computing resources which can result in high variability of service quality. We consider the problem of computation offloading over unknown edge cloud networks with a sequence of timely computation jobs. Motivated by the MapReduce computation paradigm, we assume that each computation job can be partitioned to smaller Map functions which are processed at the edge, and the Reduce function is computed at the user after the Map results are collected from the edge nodes. We model the service quality of each edge device as function of context. The user decides the computations to offload to each device with the goal of receiving a recoverable set of computation results in the given deadline. By leveraging the *coded computing* framework in order to tackle failures or stragglers in computation, we formulate this problem using contextual-combinatorial multi-armed bandits (CC-MAB), and aim to maximize the cumulative expected reward. We propose an online learning policy called *online coded edge computing policy*, which provably achieves asymptotically-optimal performance in terms of regret loss compared with the optimal offline policy for the proposed CC-MAB problem. In terms of the cumulative reward, it is shown that the online coded edge computing policy significantly outperforms other benchmarks via numerical studies.**

***Index Terms*— Edge computing, coded computing, online learning, multi-armed bandits.**

Manuscript received May 7, 2020; revised December 8, 2020; accepted January 17, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Iosifidis. Date of publication February 24, 2021; date of current version June 16, 2021. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract HR001117C0053, in part by the Army Research Office (ARO) under Award W911NF1810400, in part by NSF under Grant CCF-1703575, Grant CCF-1763673, Grant CNS-2003035, and Grant CNS-2002874, in part by the Office of Naval Research (ONR) under Award N00014-16-1-2189, in part by UC Office of President under Grant LFR-18-548175, and in part by Intel. A preliminary part of this work was presented in IEEE ISIT 2020. *(Corresponding author: Chien-Sheng Yang.)*

Chien-Sheng Yang and A. Salman Avestimehr are with the Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: chienshy@usc.edu; avestimehr@ee.usc.edu).

Ramtin Pedarsani is with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: ramtin@ece.ucsb.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2021.3058685, provided by the authors.

Digital Object Identifier 10.1109/TNET.2021.3058685

I. INTRODUCTION

RECENT advancements in edge cloud has enabled users to offload their computations of interest to the edge for processing. Specifically, there has been a significant increase in utilizing the edge cloud for event-driven and time-sensitive computations (e.g., IoT applications and cognitive services), in which the users increasingly demand timely services with deadline constraints, i.e., computations of requests have to be finished within specified deadlines. However, large-scale distributed computing networks can substantially suffer from unpredictable and unreliable computing infrastructure which can result in high variability of computing resources, i.e., service quality of the computing resources may vary over time. The speed variation has several causes including hardware failure, co-location of computation tasks, communication bottlenecks, etc [2], [3]. While edge computing has offered a novel framework for computing service provisioning, a careful design of task scheduling policy is still needed to guarantee the timeliness of task processing due to the increasing demand on real-time response of various applications and the unknown environment of the network.

To take advantage of the parallel computing resources for reducing the total latency, the applications are often modeled as a MapReduce computation model, i.e., the computation job can be partitioned to some smaller Map functions which can be distributedly processed by the edge devices. Since the data transmissions between the edge devices can result in large latency delay, it is often the case that the user computes the Reduce function on the results of the Map functions upon receiving the computation results of edge devices to complete the computation job.

In this article, we study the problem of computation offloading over edge cloud networks with particular focus on *unknown environment of computing resources* and *timely computation jobs*. We consider a dynamic computation model, where a sequence of computation jobs needs to be computed over the (encoded) data that is distributedly stored at the edge nodes. More precisely, in an online manner, computation jobs with given deadlines are submitted to the edge network, i.e., each computation has to be finished within the given deadline. We assume the service quality (success probability of returning results back to the user in deadline) of each edge device is parameterized by a context (collection of factors

that affect each edge device). The user aims at selecting edge devices from the available edge devices such that the user can receive a recoverable set of computation results in the given deadline. Our goal is then to design an efficient edge computing policy that maximizes the cumulative expected reward, where the expected reward collected at each round is a linear combination of the success probability of the computation and the amount of computational resources used (with negative sign).

One significant challenge in this problem is the joint design of (1) data storage scheme to provide robustness against unknown behaviors of edge devices; (2) computation offloading to edge device; and (3) an online learning policy for making the offloading decisions based on the past observed events. In our model, the computation capacities of the devices (e.g., how likely the computation can be returned to the user within the deadline) are unknown to the user.

As the main contributions of the article, we introduce a *coded computing* framework in which the data is encoded and stored at the edge devices in order to provide robustness against unknown computation capabilities of the devices. The key idea of coded computing is to encode the data and design each worker's computation task such that the fastest responses of any k workers out of total of n workers suffice to complete the distributed computation, similar to classical coding theory where receiving any k symbols out of n transmitted symbols enables the receiver to decode the sent message. Under coded computing framework, we formulate a contextual-combinatorial multi-armed bandit (CC-MAB) problem for the edge computing problem, in which the Lagrange coding scheme is utilized for data encoding [4].

Then, we propose a policy called *online coded edge computing policy*, and show that it achieves asymptotically optimal performance in terms of regret loss compared with the optimal offline policy for the proposed CC-MAB problem by the careful design of the policy parameters. To prove the asymptotic optimality of online coded edge computing policy, we divide the expected regret to three regret terms due to (1) exploration phases, (2) bad selections of edge devices in exploitation phases, and (3) good selections of edge devices in exploitation phases; then we bound these three regrets separately.

In addition to proving the asymptotic optimality of online coded edge computing policy, we carry out numerical studies using the real world scenarios of Amazon EC2 clusters. In terms of the cumulative reward, the results show that the online coded edge computing policy significantly outperforms other benchmarks.

In the following, we summarize the key contributions in this article:

- We formulate the problem of coded edge computing using the CC-MAB framework.
- We propose online coded edge computing policy, which is provably asymptotically optimal.
- We show that the online coded edge computing policy outperforms other benchmarks via numerical studies.

A. Related Prior Work

Next, we provide a brief literature review that covers three main lines of work: task scheduling over cloud networks, coded computing, and the multi-armed bandit problem.

In the dynamic task scheduling problem, jobs arrive to the network according to a stochastic process, and get scheduled dynamically over time. The first goal in task scheduling is to find a throughput-optimal scheduling policy (see e.g., [5]), i.e. a policy that stabilizes the network, whenever it can be stabilized. For example, Max-Weight scheduling, first proposed in [6], [7], is known to be throughput-optimal for wireless networks, flexible queueing networks [8], data centers networks [9] and dispersed computing networks [10]. Moreover, there have been many works which focus on task scheduling problem with deadline constraints over cloud networks (see e.g., [11]).

Coded computing broadly refers to a family of techniques that utilize coding to inject computation redundancy in order to alleviate the various issues that arise in large-scale distributed computing. In the past few years, coded computing has had a tremendous success in various problems, such as straggler mitigation and bandwidth reduction (e.g., [12]–[21]). Coded computing has also been expanded in various directions, such as heterogeneous networks (e.g., [22]), partial stragglers (e.g., [23]), secure and private computing (e.g., [4], [24]–[27]), distributed optimization (e.g., [28]), federated learning (e.g., [29]–[31]), blockchains (e.g., [32], [33]). In a dynamic setting, [34], [35] consider the coded computing framework with deadline constraints and develops a learning strategy that can adaptively assign computation loads to cloud devices. In this article, we go beyond the two states Markov model considered in [34], [35], and make a substantial progress by combining the ideas of coded computing with contextual-combinatorial MAB, which is a more general framework that does not make any strong assumption (e.g., Markov model) on underlying model for the speed of edge devices.

The multi-armed bandit (MAB) problem has been widely studied to address the critical tradeoff between exploration and exploitation in sequential decision making under uncertainty of environment [36]. The goal of MAB is to learn the single optimal arm among a set of candidate arms of a priori unknown rewards by sequentially selecting one arm each time and observing its realized reward [37]. Contextual bandit problem extends the basic MAB by considering the context-dependent reward functions [38]–[40]. The combinatorial bandit problem is another extension of the MAB by allowing multiple-play (select a set of arms) each time [41], [42]. The contextual-combinatorial MAB problem considered in this article has also received much attention recently [43]–[46]. However, [44], [46] assume that the reward of an action is a linear function of the contexts different from the reward function considered in our article. [45] assumes the arm set is fixed throughout the time but the arms (edge devices) may appear and disappear across the time in edge networks.

[43] considers a CC-MAB problem for the vehicle cloud computing, in which the tasks are deadline-constrained. How-

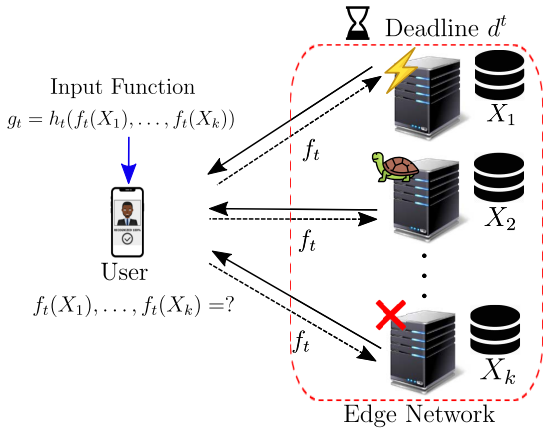

Fig. 1. Overview of online computation offloading over an edge network with timely computation requests. In round t, the goal of user is to compute the Map functions $f_t(X_1), \ldots, f_t(X_k)$ by the deadline d^t using the edge devices.

ever, the task replication technique used in [43] is to replicate the "whole job" to multiple edge devices without taking advantage of parallelism of computational resources. Coded computing is a more general technique which allows the dataset to be first partitioned to smaller datasets and then encoded such that each device has smaller computation compared to [43]. Moreover, the success probability term (for receiving any k results out of n results) of reward function considered in our paper is more general than the success probability term (for receiving any 1 result out of n results) of reward function considered in [43].

II. System Model

A. Computation Model

We consider an edge computing problem, in which a user offloads its computation to an edge network in an online manner, and the computation is executed by the edge devices. In particular, there is a given deadline for each round of computation, i.e., computation has to be finished within the given deadline.

As shown in Fig. 1, the considered edge network is composed of a user node and a set of edge devices. There is a dataset $X_1, X_2, \ldots, X_k$ where each X_j is an element in a vector space $\mathbb{V}$ over a sufficiently large finite field $\mathbb{F}$. Each edge device prestores the data which can be possibly a function of $X_1, X_2, \ldots, X_k$.

Let $\{1, 2, \ldots, T\}$ be the index of the user's computation jobs received by the edge network over T time slots. In each round t (or time slot in a discrete-time system), the user has a computation job denoted by function g_t. Especially, we assume that function g_t can be computed by

$$g_t(X_1, X_2, \ldots, X_k) = h_t(f_t(X_1), f_t(X_2), \ldots, f_t(X_k))$$

where function g_t and f_t (with degree $\deg(f_t)$) are multivariate polynomial functions with vector coefficients. In such edge network and motivated by a MapReduce setting, the user is interested in computing Map functions $f_t(X_1), f_t(X_2), \ldots, f_t(X_k)$ in each round t and the user computes Reduce function h_t on those results of Map functions to obtain $g_t(X_1, X_2, \ldots, X_k)$.

Remark 1: We note that the considered computation model naturally appears in many machine learning applications which use gradient-type algorithms. For example, in linear regression problems given $\vec{y}_j$ which is the vector of observed labels for data X_j, each worker j computes $f_t(X_j) = X_j^\top(X_j\vec{w}_t - \vec{y}_j)$ which is the gradient of the quadratic loss function $\frac{1}{2}\|X_j\vec{w}_t - \vec{y}_j\|^2$ with respect to the weight vector $\vec{w}_t$ in round t. To complete the update $\vec{w}_{t+1} = g_t(X_1, \ldots, X_k) = \vec{w}_t - \beta_t \sum_{j=1}^k f_t(X_j)$, the user has to collect the computation results $f_t(X_1), f_t(X_2), \ldots f_t(X_k)$.

Moreover, the considered computation model also holds for various edge computing applications. For example, in a mobile navigation application, the goal of user is to compute the fastest route to its destination. Given a dataset containing the map information and the traffic conditions over a period of time, edge devices compute map functions which output all possible routes between the two end locations. After collecting the intermediate results from edge devices, the user computes the best route.

B. Network Model

In an edge computing network, whether a computation result can be returned to the user depends on many factors. For example, the computation load of an edge device influences its runtime; the output size of the computation task affects the transmission delay, etc. Such factors are referred to as *context* throughout the article. The impact of each context on the edge devices is unknown to the user. More specifically, the computation service of each edge device is modeled as follows.

Let Φ_T be the context space of dimension D_T includes D_T different information of computation task, e.g., size of input/output, size of computation, and deadline, etc. Let Φ_S be the context space of dimension D_S for edge devices which includes the information related to edge devices such as computation speed, bandwidth, etc. Let $\Phi = \Phi_T \times \Phi_S$ be the joint context space which is assumed to be bounded and thus can be defined by $\Phi = [0, 1]^D$ and $D = D_T + D_S$ is the dimension of context space Φ without loss of generality.

In each round t, let $\mathcal{V}^t$ denote the set of edge devices available to the user for computation, i.e., the available set of devices might change over time. Moreover, we denote by b^t the budget (maximum number of devices to be used) in round t. The service delay (computation time plus transmission time) of each edge device ν is parameterized by a given context $\phi_\nu^t \in \Phi$. We denote by c_ν^t the service delay of edge device ν, and d^t the computation deadline in round t. Let $q_\nu^t = \mathbb{1}_{\{c_\nu^t \le d^t\}}$ be the indicator that the service delay of edge device ν is smaller than or equal to the given deadline d^t in round t. Also, let $\mu(\phi_\nu^t) = \mathbb{E}[q_\nu^t] = \mathbb{P}(c_\nu^t \le d^t)$ be the success probability that edge device ν returns the computation result back to the user within deadline d^t, and $\boldsymbol{\mu}^t = \{\mu(\phi_\nu^t)\}_{\nu \in \mathcal{V}^t}$ be the collection of success probabilities of edge devices in round t. Let us illustrate the model through a simple example.

Example 1: In [22], the shifted exponential distributions have been demonstrated to be a good fit for modeling the execution time of a node in cloud networks. Thus, we can

model the success probability of an edge device as follows:

$$\mu(\phi^t) = \mathbb{P}(c^t \leq d^t) = \begin{cases} 1 - e^{-\lambda^t (d^t - a^t)} & , \ d^t \geq a^t \\ 0 & , \ a^t > d^t \geq 0, \end{cases}$$

where the context space Φ consists of the deadline d^t, the shift parameter $a^t > 0$, and the straggling parameter $\lambda^t > 0$ associated with an edge device.

C. Problem Statement

Let $\mathcal{V} = \{1, 2, \ldots, |\mathcal{V}|\}$ be the set of all edge devices in the network. Given context $\boldsymbol{\phi}^t = \{\phi_\nu^t\}_{\nu \in \mathcal{V}^t}$ of the edge devices available to the user in round t, the goal of the user is to select a subset of edge devices from the available set of edge devices $\mathcal{V}^t \subseteq \mathcal{V}$, and decide what to be computed by each selected edge device, such that a *recoverable* (or decodable as will be clarified later) set of computation results $f_t(X_1), \ldots f_t(X_k)$ can be returned to the user within deadline d^t.

III. Online Coded Edge Computing

In this section, we introduce a coded computing framework for the edge computing problem, and formulate the problem as a contextual-combinatorial multi-armed bandit (CC-MAB) problem. Then, we propose a policy called *online coded edge computing policy*, which is a context-aware learning algorithm.

A. Lagrange Coded Computing

For the data storage of edge devices, we leverage a linear coding scheme called the Lagrange coding scheme [4] which is demonstrated to simultaneously provide resiliency, security, and privacy in distributed computing. We start with an illustrative example.

In each round t, we consider a computation job which consists of computing quadratic functions $f_t(X_j) = X_j^\top (X_j \vec{w}_t - \vec{y}_j)$ over available edge devices $\mathcal{V}^t = \{1, 2, \ldots, 6\}$, where input dataset X is partitioned to X_1, X_2. Then, we define function m as follows:

$$m(z) \triangleq X_1 \frac{z-1}{0-1} + X_2 \frac{z-0}{1-0} = z(X_2 - X_1) + X_1, \quad (1)$$

in which $m(0) = X_1$ and $m(1) = X_2$. Then, we encode X_1 and X_2 to $\tilde{X}_\nu = m(\nu - 1)$, i.e., $\tilde{X}_1 = X_1$, $\tilde{X}_2 = X_2$, $\tilde{X}_3 = -X_1 + 2X_2$, $\tilde{X}_4 = -2X_1 + 3X_2$, $\tilde{X}_5 = -3X_1 + 4X_2$ and $\tilde{X}_6 = -4X_1 + 5X_2$. Each edge device $\nu \in \{1, 2, \ldots, 6\}$ prestores an encoded data chunk $\tilde{X}_\nu$ locally. If edge device ν is selected in round t, it computes $f_t(\tilde{X}_\nu) = \tilde{X}_\nu^\top (\tilde{X}_\nu \vec{w}_t - \vec{y}_\nu)$ and returns the result back to the user upon its completion. We note that $f_t(\tilde{X}_\nu) = f_t(m(\nu - 1))$ is an evaluation of the composition polynomial $f_t(m(z))$, whose degree at most 2, which implies that $f_t(m(z))$ can be recovered by any 3 results via polynomial interpolation. Then we have $f_t(X_1) = f_t(m(0))$ and $f_t(X_2) = f_t(m(1))$.

Formally, we describe Lagrange coding scheme as follows:

We first select k distinct elements $\beta_1, \beta_2, \ldots, \beta_k$ from $\mathbb{F}$, and let m be the respective *Lagrange interpolation polynomial*

$$m(z) \triangleq \sum_{j=1}^{k} X_j \prod_{l \in [k] \setminus \{j\}} \frac{z - \beta_l}{\beta_j - \beta_l}, \quad (2)$$

where $u : \mathbb{F} \to \mathbb{V}$ is a polynomial of degree $k-1$ such that $m(\beta_j) = X_j$. Recall that $\mathcal{V} = \cup_{t=1}^{T} \mathcal{V}^t$ which is the set of all edge devices. To encode input $X_1, X_2, \ldots, X_k$, we select $|\mathcal{V}|$ distinct elements $\alpha_1, \alpha_2, \ldots, \alpha_{|\mathcal{V}|}$ from $\mathbb{F}$, and encode $X_1, X_2, \ldots, X_k$ to $\tilde{X}_v = m(\alpha_v)$ for all $v \in [|\mathcal{V}|]$, i.e.,

$$\tilde{X}_v = m(\alpha_v) \triangleq \sum_{j=1}^{k} X_j \prod_{l \in [k] \setminus \{j\}} \frac{\alpha_v - \beta_l}{\beta_j - \beta_l}. \quad (3)$$

Each edge device $\nu \in \mathcal{V}$ stores $\tilde{X}_\nu$ locally. If edge device ν is selected in round t, it computes $f_t(\tilde{X}_\nu)$ and returns the result back to the user upon its completion. Then, the optimal recovery threshold Y^t using Lagrange coding scheme is

$$Y^t = (k-1)\deg(f_t) + 1 \quad (4)$$

which guarantees that the computation tasks $f_t(X_1), \ldots, f_t(X_k)$ can be recovered when the user receives any Y^t results from the edge devices. The encoding of Lagrange coding scheme is oblivious to the computation task f_t. Also, decoding and encoding process in Lagrange coding scheme rely on polynomial interpolation and evaluation which can be done efficiently.

Remark 2: We note that the data newly generated in edge device can be encoded and distributed to other devices at off-peak time. Especially, a key property of LCC is that the encoding process can be done incrementally, i.e., when there are some new added datasets, the update of encoded data can be done incrementally by encoding only on the new data instead of redoing the encoding on all the datasets. For example, let us consider the case that each data X_j is represented by a vector. When there is a new generated data element x_j added to each data X_j, we just encode new data elements $x_1, x_2, \ldots, x_k$ to $\tilde{x}_1, \tilde{x}_2 \ldots$ and the new encoded data can be obtained by appending the new encoded data to old encoded data vectors $\tilde{X}_1, \tilde{X}_2, \ldots$.

B. CC-MAB for Coded Edge Computing

Now we consider a coded computing framework in which the Lagrange coding scheme is used for data encoding, i.e., each edge device ν prestores encoded data $\tilde{X}_\nu$. The encoding process is only performed once for dataset $X_1, \ldots, X_k$. After Lagrange data encoding, the size of input data and computation of each user do not change, i.e., context ϕ_ν^t of each edge device ν remains the same.

More specifically, we denote by $\mathcal{A}^t$ the set of devices which are selected in round t for computation. In each round t, the user picks a subset of devices $\mathcal{A}^t$ from all available devices $\mathcal{V}^t$, and we call $\mathcal{A}^t \subseteq \mathcal{V}^t$ the "offloading decision". The reward function $r(\mathcal{A}^t)$ achieved by offloading decision $\mathcal{A}^t$ is composed of the reward term and the cost term, which is defined as follows:

$$r(\mathcal{A}^t) = \begin{cases} 1 - \eta |\mathcal{A}^t|, & \text{if } \sum_{\nu \in \mathcal{A}^t} q_\nu^t \geq Y^t \\ -\eta |\mathcal{A}^t|, & \text{if } \sum_{\nu \in \mathcal{A}^t} q_\nu^t < Y^t \end{cases} \quad (5)$$

where the term $|\mathcal{A}^t|$ captures the cost of using offloading decision $\mathcal{A}^t$ with the unit cost η for using one edge device, and Y^t is the optimal recovery threshold defined in (4). More

precisely, the reward term is equal to 1 if the total number of received results is greater than the optimal recovery threshold, i.e., $\sum_{\nu\in\mathcal{A}^t} q_\nu^t \geq Y^t$; otherwise the reward term is equal to 0. On the other hand, the cost term is defined as $-\eta|\mathcal{A}^t|$ which is the cost of using $\mathcal{A}^t$.

Then, the expected reward denoted by $u(\boldsymbol{\mu}^t, \mathcal{A}^t)$ in round t can be rewritten as follows:

$$u(\boldsymbol{\mu}^t, \mathcal{A}^t) = \sum_{s=Y^t}^{|\mathcal{A}^t|} \sum_{\mathcal{A}\subseteq\mathcal{A}^t, |\mathcal{A}|=s} \prod_{\nu\in\mathcal{A}} \mu(\phi_\nu^t) \prod_{\nu\in\mathcal{A}^t\setminus\mathcal{A}} (1-\mu(\phi_\nu^t)) - \eta|\mathcal{A}^t| \quad (6)$$

where the first term of the expected reward of an offloading decision is the success probability that there are at least Y^t computation results received by the user for LCC decoding.

Consider an arbitrary sequence of computation jobs indexed by $\{1, 2, \ldots, T\}$ for which the user makes offloading decisions $\{\mathcal{A}^t\}_{t=1}^T$. To maximize the expected cumulative reward, we introduce a contextual-combinatorial multi-armed bandit (CC-MAB) problem for coded edge computing defined as follows:

1) CC-MAB for Coded Edge Computing:

$$\max_{\{\mathcal{A}^t\}_{t=1}^T} \sum_{t=1}^T u(\boldsymbol{\mu}^t, \mathcal{A}^t) \quad (7)$$

$$\text{s.t. } \mathcal{A}^t \subseteq \mathcal{V}^t,\ |\mathcal{A}^t| \leq b^t,\ \forall t \in [T] \quad (8)$$

where the constraint (8) indicates that the number of edge devices in $\mathcal{A}^t$ cannot exceed the budget b^t in round t. The proposed CC-MAB problem is equivalent to solving an independent subproblem in each round t as follows:

$$\max_{\mathcal{A}^t} \sum_{s=Y^t}^{|\mathcal{A}^t|} \sum_{\mathcal{A}\subseteq\mathcal{A}^t, |\mathcal{A}|=s} \prod_{\nu\in\mathcal{A}} \mu(\phi_\nu^t) \prod_{\nu\in\mathcal{A}^t\setminus\mathcal{A}} (1-\mu(\phi_\nu^t)) - \eta|\mathcal{A}^t|$$
$$\text{s.t. } \mathcal{A}^t \subseteq \mathcal{V}^t;\ |\mathcal{A}^t| \leq b^t.$$

Remark 3: We note that the proposed CC-MAB not only works for LCC but also for any other coding schemes. In this article, we focus on LCC since LCC is a universal and optimal encoding technique for arbitrary multivariate polynomial computations.

C. Optimal Offline Policy

We now assume that the success probability of each edge device $\nu \in \mathcal{V}^t$ is known to the user. In round t, to find the optimal $\mathcal{A}^{t*}$, we present the following intuitive lemma proved in Appendix E.

Lemma 1: Without loss of generality, we assume $\mu(\phi_1^t) \geq \mu(\phi_2^t) \geq \cdots \geq \mu(\phi_{|\mathcal{V}^t|}^t)$ in round t. Considering all possible sets $\mathcal{A}_g^t \subseteq \mathcal{V}^t$ with fixed cardinality n_g, the optimal $\mathcal{A}_g^{t*}$ with cardinality n_g that achieves the largest expected reward $u(\boldsymbol{\mu}^t, \mathcal{A}_g^t)$ is

$$\mathcal{A}_g^{t*} = \{1, 2, \ldots, n_g\} \quad (9)$$

which represents the set of n_g edge devices having largest success probability $\mu(\phi_\nu^t)$ among all the edge devices.

Algorithm 1: Optimal Offline Policy

Input: $\mathcal{V}^t, b^t, Y^t, \mu(\phi_\nu^t), \nu \in \mathcal{V}^t$;
Initialization: $\mathcal{A} = \emptyset$, $\mathcal{A}_{\text{opt}} = \emptyset$, $u_{\text{opt}} = 0$;
Sort $\boldsymbol{\mu}^t : \mu(\phi_1^t) \geq \mu(\phi_2^t) \geq \cdots \geq \mu(\phi_{|\mathcal{V}^t|}^t)$;
$\mathcal{A} \leftarrow \{1, 2, \ldots, Y^t\}$;
$\mathcal{A}_{\text{opt}} \leftarrow \{1, 2, \ldots, Y^t\}$;
$u_{\text{opt}} \leftarrow u(\boldsymbol{\mu}^t, \mathcal{A})$;
for $z \leftarrow Y^t + 1$ **to** b^t **do**
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{z\}$;
  if $u(\boldsymbol{\mu}^t, \mathcal{A}) > u_{opt}$ **then**
    $\mathcal{A}_{\text{opt}} \leftarrow \mathcal{A}$;
    $u_{\text{opt}} \leftarrow u(\boldsymbol{\mu}^t, \mathcal{A})$
  end
end
return $\mathcal{A}_{\text{opt}}$

By Lemma 1, to find the optimal set $\mathcal{A}^{t*}$, we can only focus on finding the optimal size of $\mathcal{A}^t$. Since there are only b^t choices for size of $|\mathcal{A}^t|$ (i.e., $1, 2, \ldots, b^t$), this procedure can be done by a linear search with the complexity linear in the number of edge devices $|\mathcal{V}^t|$. We present the optimal offline policy in Algorithm 1.

Remark 4: We note that the expected reward function considered in [43] is a submodular function, which can be maximized by a greedy algorithm. However, the expected reward function defined in (6) *is more general which cannot be maximized by the greedy algorithm. More specifically, one can show that the expected reward defined in equation* (6) *is not submodular by checking the property of submodular functions, i.e., for all possible subsets $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$, $u(\boldsymbol{\mu}, \{\nu\} \cup \mathcal{A}) - u(\boldsymbol{\mu}, \mathcal{A}) \geq u(\boldsymbol{\mu}, \{\nu\} \cup \mathcal{B}) - u(\boldsymbol{\mu}, \mathcal{B})$ does not hold. Without the property of submodularity, Lemma 1 enables us to maximize equation* (6) *by a linear search.*

Let $\{\mathcal{A}^t\}_{t=1}^T$ be the offloading decisions derived by a certain policy. The performance of this policy is evaluated by comparing its loss with respect to the optimal offline policy. This loss is called the regret of the policy which is formally defined as follows:

$$R(T) = \mathbb{E}\big[\sum_{t=1}^T r(\mathcal{A}^{t*}) - r(\mathcal{A}^t)\big] \quad (10)$$

$$= \sum_{t=1}^T u(\boldsymbol{\mu}^t, \mathcal{A}^{t*}) - u(\boldsymbol{\mu}^t, \mathcal{A}^t). \quad (11)$$

In general, the user does not know in advance the success probabilities of edge devices due to the uncertainty of the environment of edge network. In the following subsection, we will propose an online learning policy for the proposed CC-MAB problem which enables the user to learn the success probabilities of edge devices over time by observing the service quality of each selected edge device, and then make offloading decisions adaptively.

D. Online Coded Edge Computing Policy

Now, we describe the proposed online edge computing policy. The proposed policy has two parameters h_T and $K(t)$

Algorithm 2: Online Coded Edge Computing Policy

Input: $T, h_T, K(t)$;
Initialization: $\mathcal{P}_T$; $C(p)=0$, $\hat{\mu}(p)=0$, $\forall p \in \mathcal{P}_T$;
for $t \leftarrow 1$ **to** T **do**
  Observe edge device $\mathcal{V}^t$ and contexts $\boldsymbol{\phi}^t$;
  Find $\boldsymbol{p}^t=\{p_\nu^t\}_{\nu\in\mathcal{V}^t}$, $p_\nu^t \in \mathcal{P}_T$ such that $\phi_\nu^t \in p_\nu^t$;
  Identify $\mathcal{P}^{ue,t}$ and $\mathcal{V}^{ue,t}$;
  if $\mathcal{P}^{ue,t} \neq \emptyset$ **then**
    if $|\mathcal{V}^{ue,t}| \geq b^t$ **then**
      $\mathcal{A}^t \leftarrow$ randomly pick b^t edge devices in $\mathcal{V}^{ue,t}$;
    else
      $\mathcal{A}^t \leftarrow$ pick all edge devices in $\mathcal{V}^{ue,t}$ and other $(b^t-|\mathcal{V}^{ue,t}|)$ ones with the largest $\hat{\mu}(p_\nu^t)$ in $\mathcal{V}^t \backslash \mathcal{V}^{ue,t}$
    end
  else
    $\mathcal{A}^t \leftarrow$ obtained by Algorithm 1 based on $\hat{\boldsymbol{\mu}}^t$ and b^t
  end
  for *each edge device* $\nu \in \mathcal{A}^t$ **do**
    Observe q_ν^t of edge device ν;
    Update $\hat{\mu}(p_\nu^t)=\frac{\hat{\mu}(p_\nu^t)C(p_\nu^t)+q_\nu^t}{C(p_\nu^t)+1}$;
    Update $C(p_\nu^t)=C(p_\nu^t)+1$;
  end
end

to be designed, where h_T decides how we partition the context space, and $K(t)$ is a deterministic and monotonically increasing function, used to identify the under-explored context. The proposed online coded edge computing policy (see Algorithm 2) is performed as follows:

Initialization Phase: Given parameter h_T, the proposed policy first creates a partition denoted by $\mathcal{P}_T$ for the context space Φ, which splits Φ into $(h_T)^D$ sets. Each set is a D-dimensional hypercube of size $\frac{1}{h_T} \times \cdots \times \frac{1}{h_T}$. For each hypercube $p \in \mathcal{P}_T$, the user keeps a counter $C^t(p)$ which is the number of selected edge devices that have context ϕ_ν^t in hypercube p before round t. Moreover, the policy also keeps an estimated success probability denoted by $\hat{\mu}^t(p)$ for each hypercube p. Let $\mathcal{Q}^t(p)=\{q_\nu^\tau : \phi_\nu^\tau \in p, \nu \in \mathcal{A}^\tau, \tau=1,\ldots,t-1\}$ be the set of observed indicators (successful or not) of edge devices with context in p before round t. Then, the estimated success probability for edge devices with context $\phi_\nu^t \in p$ is computed by $\hat{\mu}^t(p)=\frac{1}{C^t(p)}\sum_{q\in\mathcal{Q}^t(p)} q$.

In each round t, the proposed policy has the following phases:

Hypercube Identification Phase: Given the contexts of all available edge devices $\boldsymbol{\phi}^t=\{\phi_\nu^t\}_{\nu\in\mathcal{V}^t}$, the policy determines the hypercube $p_\nu^t \in \mathcal{P}_T$ for each context ϕ_ν^t such that ϕ_ν^t is in p_ν^t. We denote by $\boldsymbol{p}^t=\{p_\nu^t\}_{\nu\in\mathcal{V}^t}$ the collection of these identified hypercubes in round t. To check whether there exist hypercubes $p \in \boldsymbol{p}^t$ that have not been explored sufficiently, we define the under-explored hypercubes in round t as follows:

$$\mathcal{P}_T^{ue,t}=\{p \in \mathcal{P}_T : \exists \nu \in \mathcal{V}^t, \phi_\nu^t \in p, C^t(p) \leq K(t)\}. \quad (12)$$

Also, we denote by $\mathcal{V}^{ue,t}$ the set of edge devices which fall in the under-explored hypercubes, i.e., $\mathcal{V}^{ue,t}=\{\nu \in \mathcal{V}^t : p_\nu^t \in \mathcal{P}_T^{ue,t}\}$. Depending on $\mathcal{V}^{ue,t}$ in round t, the proposed policy then either enters an exploration phase or an exploitation phase.

Exploration Phase: If $\mathcal{V}^{ue,t}$ is non-empty, the policy enters an exploration phase. If set $\mathcal{V}^{ue,t}$ contains at least b^t edge devices (i.e., $|\mathcal{V}^{ue,t}| \geq b^t$), then the policy randomly selects b^t edge devices from $\mathcal{V}^{ue,t}$. If $\mathcal{V}^{ue,t}$ contains fewer than b^t edge devices ($|\mathcal{V}^{ue,t}| < b^t$), then the policy selects all edge devices from $\mathcal{V}^{ue,t}$. To fully utilize the budget b^t, the remaining $(b^t-|\mathcal{V}^{ue,t}|)$ ones are picked from the edge devices with the highest estimated success probability among the remaining edge devices in $\mathcal{V}^t \backslash \mathcal{V}^{ue,t}$.

Exploitation Phase: If $\mathcal{V}^{ue,t}$ is empty, the policy enters an exploitation phase and it selects $\mathcal{A}^t$ using the optimal offline policy based on the estimated success probabilities $\hat{\boldsymbol{\mu}}^t=\{\hat{\mu}^t(p_\nu^t)\}_{\nu\in\mathcal{V}^t}$.

Update Phase: After selecting the edge devices, the proposed policy observes whether each selected edge device returns the result within the deadline; then, it updates $\hat{\mu}^t(p_\nu^t)$ and $C^t(p_\nu^t)$ of each hypercube $p_\nu^t \in \mathcal{P}_T$.

The following example illustrates how the policy works given parameters h_T and $K(t)$.

Example 2: Consider the edge computing network in which the success probability of an edge device is defined by a shifted exponential distribution as defined in Example 1. It can be shown that the Hölder condition with $\alpha=1$ holds. Then, we have parameters $h_T=\lceil T^{\frac{1}{6}} \rceil$ and $K(t)=t^{\frac{1}{3}}\log(t)$. We assume that the online coded edge computing policy is run over time horizon $T=1000$. Then, we have $h_T=4$. Before running the policy, we create $\mathcal{P}_T$ by partitioning the domain of each context (i.e., deadlines, shift parameters and straggling parameters) into $h_T=4$ intervals, which generates totally 64 sets. We keep a counter $C^t(p)$ for each generated hypercube $p \in \mathcal{P}_T$. In hypercube identification phase, if there exists edge device ν with ϕ_ν^t such that ϕ_ν^t is located in hypercube p and the counter $C^t(p)$ is smaller than $K(t)$, then p is the under-explored hypercube. The policy will proceed to either exploration phase or exploitation phase depending on whether under-explored hypercube exists.

IV. Asymptotic Optimality of Online Coded Edge Computing Policy

In this section, by providing the design of policy parameters h_T and $K(t)$, we show that the online coded edge computing policy achieves a sublinear regret in the time horizon T which guarantees an asymptotically optimal performance, i.e., $\lim_{T\to\infty}\frac{R(T)}{T}=0$.

To conduct the regret analysis for the proposed CC-MAB problem, we make the following assumption on the success probabilities of edge devices in which the devices' success probabilities are equal if they have the same contexts. This natural property is formalized by the Hölder condition defined as follows:

Assumption 1 (Hölder Condition): A real function f on D-dimensional Euclidean space satisfies a Hölder condition, when there exist $L>0$ and $\alpha>0$ for any two contexts $\phi, \phi' \in \Phi$, such that $|f(\phi)-f(\phi')| \leq L \parallel \phi-\phi' \parallel^\alpha$, where $\parallel \cdot \parallel$ is the Euclidean norm.

Under Assumption 1, we choose parameters $h_T = \lceil T^{\frac{1}{3\alpha+D}} \rceil$ for the partition of context space Φ and $K(t) = t^{\frac{2\alpha}{3\alpha+D}} \log(t)$ in round t for identifying the under-explored hypercubes of the context. We present the following theorem which shows that the proposed online coded edge computing policy has a sublinear regret upper bound.

Theorem 1 (Regret Upper Bound): Let $K(t) = t^{\frac{2\alpha}{3\alpha+D}} \log(t)$ *and* $h_T = \lceil T^{\frac{1}{3\alpha+D}} \rceil$. *If the Hölder condition holds, the regret* $R(T)$ *is upper-bounded as follows:*

$$\begin{aligned} R(T) \leq (1+\eta B)2^D (T^{\frac{2\alpha+D}{3\alpha+D}} \log(T) + T^{\frac{D}{3\alpha+D}}) \\ +(1+\eta B)B\frac{\pi^2}{3}\sum_{k=1}^{B}\binom{|\mathcal{V}|}{k} \\ +(3LD^{\frac{\alpha}{2}} + \frac{6\alpha+2D}{2\alpha+D})BMT^{\frac{2\alpha+D}{3\alpha+D}}, \end{aligned}$$

where $B = \max_{1\leq t\leq T} b^t$ *and* $M = \max_{1\leq t\leq T}\binom{B-1}{Y^t-1}$. *The dominant order of the regret* $R(T)$ *is* $O(T^{\frac{2\alpha+D}{3\alpha+D}} \log(T))$ *which is sublinear to* T.

Proof: We first define the following terms. For each hypercube $p \in \mathcal{P}_T$, we define $\overline{\mu} = \sup_{\phi\in p}\mu(\phi)$ and $\underline{\mu} = \inf_{\phi\in p}\mu(\phi)$ as the best and worst success probabilities over all contexts $\phi \in p$. Also, we define the context at center of a hypercube p as $\tilde{\phi}_p$ and its success probability $\tilde{\mu}(p) = \mu(\tilde{\phi}_p)$. Given a set of available edge devices $\mathcal{V}^t$, the corresponding context set $\Phi^t = \{\phi_\nu^t\}_{\nu\in\mathcal{V}^t}$ and the corresponding hypercube set $\mathcal{P}^t = \{p_\nu^t\}_{\nu\in\mathcal{V}^t}$ for each round t, we also define $\overline{\boldsymbol{\mu}}^t = \{\overline{\mu}(p_\nu^t)\}_{\nu\in\mathcal{V}^t}$, $\underline{\boldsymbol{\mu}}^t = \{\underline{\mu}(p_\nu^t)\}_{\nu\in\mathcal{V}^t}$ and $\tilde{\boldsymbol{\mu}}^t = \{\tilde{\mu}(p_\nu^t)\}_{\nu\in\mathcal{V}^t}$. For each round t, we define set $\tilde{\mathcal{A}}^t$ which satisfies

$$\tilde{\mathcal{A}}^t = \text{argmax}_{\mathcal{A}\subseteq\mathcal{V}^t, |\mathcal{A}|\leq b^t} u(\tilde{\boldsymbol{\mu}}^t, \mathcal{A}) \tag{13}$$

We then use set $\tilde{\mathcal{A}}^t$ to identify the set of edge device which are bad to select. We define

$$\mathcal{L}^t = \left\{G : G \subseteq \mathcal{V}^t, |G| \leq b^t, u(\underline{\boldsymbol{\mu}}^t, \tilde{\mathcal{A}}^t) - u(\overline{\boldsymbol{\mu}}^t, G) \geq At^\theta\right\}$$

to be the set of *suboptimal subsets of arms* for hypercube set $\mathcal{P}^t$, where $A > 0$ and $\theta < 0$ are the parameters which will be used later in the regret analysis. We call a subset $G \in \mathcal{L}^t$ *suboptimal* and $\mathcal{A}_{b^-}^t \backslash \mathcal{L}^t$ *near-optimal* for $\mathcal{P}^t$, where $\mathcal{A}_{b^-}^t$ denotes the subset of $\mathcal{V}^t$ with size less than b^t. Then the expected regret $R(T)$ can be divided into three summands:

$$R(T) = \mathbb{E}[R_e(T)] + \mathbb{E}[R_s(T)] + \mathbb{E}[R_n(T)], \tag{14}$$

where $\mathbb{E}[R_e(T)]$ is the regret due to exploration phases and $\mathbb{E}[R_s(T)]$ and $\mathbb{E}[R_n(T)]$ both correspond to regret in exploitation phases: $\mathbb{E}[R_s(T)]$ is the regret due to suboptimal choices, i.e., the subsets of edge devices from $\mathcal{L}_t$ are selected; $\mathbb{E}[R_n(T)]$ is the regret due to near-optimal choices, i.e., the subsets of edge devices from $\mathcal{A}_{b^-}^t \backslash \mathcal{L}^t$. In the following, we prove that each of the three summands is bounded.

First, the following lemma (see the proof in Appendix A) gives a bound for $\mathbb{E}[R_e(T)]$, which depends on the choice of two parameters z and γ.

Lemma 2 (Bound for $\mathbb{E}[R_e(T)]$*): Let* $K(t) = t^z \log(t)$ *and* $h_T = \lceil T^\gamma \rceil$, *where* $0 < z < 1$ *and* $0 < \gamma < \frac{1}{D}$. *If the algorithm is run with these parameters, the regret* $E[R_e(T)]$ *is bounded by*

$$\mathbb{E}[R_e(T)] \leq (1+\eta B)2^D (T^{z+\gamma D} \log(T) + T^{\gamma D}) \tag{15}$$

where $B = \max_{1\leq t\leq T} b^t$.

Next, the following lemma (see the proof in Appendix B) gives a bound for $\mathbb{E}[R_s(T)]$, which depends on the choice of z and γ with an additional condition of these parameters which has to be satisfied.

Lemma 3 (Bound for $\mathbb{E}[R_s(T)]$*): Let* $K(t) = t^z \log(t)$ *and* $h_T = \lceil T^\gamma \rceil$, *where* $0 < z < 1$ *and* $0 < \gamma < \frac{1}{D}$. *If the algorithm is run with these parameters, Assumption 1 holds, and the additional condition* $2BMt^{-\frac{z}{2}} \leq At^\theta$ *is satisfied for all* $1 \leq t \leq T$, *the regret* $\mathbb{E}[R_s(T)]$ *is bounded by*

$$\mathbb{E}[R_s(T)] \leq (1+\eta B)B\frac{\pi^2}{3}\sum_{k=1}^{B}\binom{|\mathcal{V}|}{k}, \tag{16}$$

where $B = \max_{1\leq t\leq T} b^t$, *and* $M = \max_{1\leq t\leq T}\binom{B-1}{Y^t-1}$.

Lastly, the following lemma (see the proof in Appendix C) gives a bound for $\mathbb{E}[R_n(T)]$, which depends on the choice of z and γ.

Lemma 4 (Bound for $\mathbb{E}[R_n(T)]$*): Let* $K(t) = t^z \log(t)$ *and* $h_T = \lceil T^\gamma \rceil$, *where* $0 < z < 1$ *and* $0 < \gamma < \frac{1}{D}$. *If the algorithm is run with these parameters and Assumption 1 holds, the regret* $\mathbb{E}[R_n(T)]$ *is bounded by*

$$\mathbb{E}[R_n(T)] \leq 3BMLD^{\frac{\alpha}{2}} T^{1-\gamma\alpha} + \frac{A}{1+\theta} T^{1+\theta}. \tag{17}$$

where $B = \max_{1\leq t\leq T} b^t$ *and* $M = \max_{1\leq t\leq T}\binom{B-1}{Y^t-1}$.

Now, let $K(t) = t^z \log(t)$ and $h_T = \lceil T^\gamma \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{D}$; let $H(t) = BMt^{-\frac{z}{2}}$. Also, we assume that Assumption 1 holds and the additional condition $2BMt^{-\frac{z}{2}} \leq At^\theta$ is satisfied for all $1 \leq t \leq T$. By Lemma 2, 3, and 4, the regret $R(T)$ is bounded as follows:

$$\begin{aligned} R(T) \leq (1+\eta B)2^D (T^{z+\gamma D} \log(T) + T^{\gamma D}) \\ +(1+\eta B)B\frac{\pi^2}{3}\sum_{k=1}^{B}\binom{|\mathcal{V}|}{k} \\ +3BMLD^{\frac{\alpha}{2}} T^{1-\alpha\gamma} + \frac{A}{1+\theta} T^{1+\theta}. \end{aligned} \tag{18}$$

Now, we select the parameters z, γ, A, θ according to the following values $z = \frac{2\alpha}{3\alpha+D} \in (0,1)$, $\gamma = \frac{1}{3\alpha+D} \in (0, \frac{1}{D})$, $\theta = -\frac{\alpha}{3\alpha+D}$ and $A = 2BM$. It is clear that condition $2BMt^{-\frac{z}{2}} \leq At^\theta$ is satisfied. Then, the regret $R(T)$ can be bounded as follows:

$$\begin{aligned} R(T) \leq (1+\eta B)2^D (T^{\frac{2\alpha+D}{3\alpha+D}} \log(T) + T^{\frac{D}{3\alpha+D}}) \\ +(1+\eta B)B\frac{\pi^2}{3}\sum_{k=1}^{B}\binom{|\mathcal{V}|}{k} \\ +(3LD^{\frac{\alpha}{2}} + \frac{6\alpha+2D}{2\alpha+D})BMT^{\frac{2\alpha+D}{3\alpha+D}}, \end{aligned} \tag{19}$$

which has the dominant order $O(T^{\frac{2\alpha+D}{3\alpha+D}} \log(T))$. □

Remark 5: Based on Assumption 1, the parameters h_T *and* $K(t)$ *are designed such that the regret achieved by the policy is sublinear as stated in Theorem 1. In the following, we provide*

some intuitions behind the choices of h_T and $K(T)$. We first assume that the parameters are chosen as $h_T = \lceil T^{\gamma} \rceil$ and $K(t) = t^z \log(t)$, in which γ and z are designed later. In the proof of Lemma 2, to bound $\mathbb{E}[R_e(T)]$, our main task is designing $T^{z+\gamma D}\log(T) + T^{\gamma D}$ to be a sublinear term. In the proof of Lemma 3, one of key steps is to bound $Pr(V_G^t, W^t)$ by the term of t^{-2} such that $\mathbb{E}[R_s(T)]$ is bounded by the term of $\sum_{t=1}^{\infty} t^{-2}$ which converges to a constant. In particular, we first bound $Pr(E_1)$ and $Pr(E_2)$ by the terms of $\exp\frac{-2t^z \log(t) H(t)^2}{B^2 M^2}$, and choose $H(t)$ to be $BMt^{-\frac{z}{2}}$. By Lemma 4, we bound $\mathbb{E}[R_n]$ by $3BMLD^{\frac{\alpha}{2}} T^{1-\alpha\gamma} + \frac{A}{1+\theta} T^{1+\theta}$ which can be sublinear by selecting the appropriate γ and z. By carefully selecting parameters $h_T = \lceil T^{\frac{1}{3\alpha+D}} \rceil$ and $K(t) = t^{\frac{2\alpha}{3\alpha+D}}$, the regret upper bound is shown to be subliear in Theorem 1.

V. Experiments

In this section, we demonstrate the impact of the online coded edge computing policy by simulation studies. In particular, we carry out extensive simulations using the shifted exponential models which have been demonstrated to be a good model for Amazon EC2 clusters [22].

Given a dataset partitioned to $X_1, X_2, \ldots, X_5$, we consider the linear regression problem using the gradient algorithm. It computes the gradient of quadratic loss function $\frac{1}{2}\|X_j \vec{w}_t - \vec{y}_j\|^2$ with respect to the weight vector $\vec{w}_t$ in round t, i.e., $f_t(X_j) = X_j^{\top}(X_j \vec{w}_t - \vec{y}_j)$ for all $1 \le j \le 5$. The computation is executed over a set of edge devices $\mathcal{V}$, where each edge device $\nu \in \mathcal{V}$ stores an encoded data chunk $\tilde{X}_\nu$ using Lagrange coding scheme. In such setting, we have the optimal recovery threshold $Y^t = 9$. The penalty parameter η is 0.01.

Motivated by the distribution model proposed in [22] for total execution time in cloud networks, we model the success probability of each edge device $\nu \in \mathcal{V}$ as a shifted exponential function defined as follows:

$$\mu(\phi_\nu^t) = \mathbb{P}(c_\nu^t \le d^t) = \begin{cases} 1 - e^{-\lambda_\nu^t (d^t - a_\nu^t)}, & d^t \ge a_\nu^t, \\ 0, & a_\nu^t > d^t \ge 0, \end{cases} \tag{20}$$

where the context of each edge device consists of the deadline d^t, the shift parameter $a_\nu^t > 0$, and the straggling parameter $\lambda_\nu^t > 0$ associated with edge device ν. Under this model, the dimension of context space D is 3. Moreover, for function μ defined in (20), it can be shown that the Hölder condition with $\alpha = 1$ holds. Thus, we run the online coded edge computing policy with parameters $h_T = \lceil T^{\frac{1}{6}} \rceil$ and $K(t) = t^{\frac{1}{3}} \log(t)$.

By the empirical analysis in [22], the instance of type `r4.2xlarge` is shown to have the shift parameter $a = 1.37$ and the straggling parameter $\lambda = 120$. And, the instance of type `r4.xlarge` has the shift parameter $a = 2$ and the straggling parameter $\lambda = 115$. Based on the real-world parameters for Amazon EC2 clusters, the deadline $d^t \in [d_{\min}, d_{\max}]$ (sec), the shift parameter $a_\nu^t \in [1.37, 2]$ (sec), and the straggling parameter $\lambda_\nu^t \in [115, 120]$ (1/sec) are chosen uniformly at random in each round t. We consider the following four scenarios for the simulations:

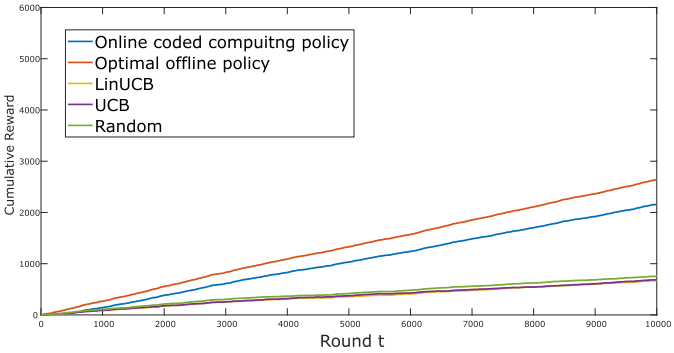

Fig. 2. Numerical evaluations for cumulative reward for Scenario 1.

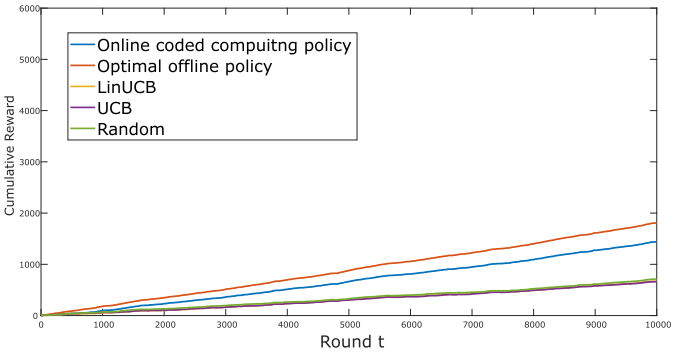

Fig. 3. Numerical evaluations for cumulative reward for Scenario 2.

- **Scenario 1:** $|\mathcal{V}| = 20$, $(d_{\min}, d_{\max}) = (1, 2)$, and $b^t = 12$.
- **Scenario 2:** $|\mathcal{V}| = 15$, $(d_{\min}, d_{\max}) = (1, 2)$, and $b^t = 12$.
- **Scenario 3:** $|\mathcal{V}| = 20$, $(d_{\min}, d_{\max}) = (1, 2)$, and $b^t = 15$.
- **Scenario 4:** $|\mathcal{V}| = 20$, $(d_{\min}, d_{\max}) = (0.5, 3)$, and $b^t = 12$.

For each scenario, the following benchmarks are considered to compare with the online coded edge computing policy:

1) **Optimal Offline policy**: Assuming knowledge of the success probability of each edge device in each round, the optimal set of edge devices is selected via Algorithm 1.
2) **LinUCB [38]**: LinUCB is a contextual-aware bandit algorithm which picks one arm in each round. We obtain a set of edge devices by repeating b^t times of LinUCB. By sequentially removing selected edge devices, we ensure that the b^t chosen edge devices are distinct.
3) **UCB [37]**: UCB algorithm is a non-contextual and non-combinatorial algorithm. Similar to LinUCB, we repeat UCB b^t times to select edge devices.
4) **Random**: A set of edge devices with size of b^t is selected randomly from the available edge devices in each round t.

Fig. 2 to Fig. 5 provide the cumulative rewards comparison of the online coded edge computing policy with the other 4 benchmarks. We make the following conclusions from Fig. 2 to Fig. 5:

- The optimal offline policy achieves the highest reward which gives an upper bound to the other policies. After a period of exploration, the proposed online policy is able to exploit the learned knowledge, and the cumulative reward approaches the upper bound.

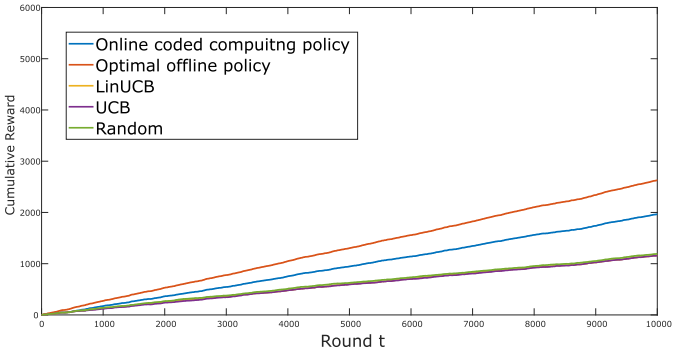

Fig. 4. Numerical evaluations for cumulative reward for Scenario 3.

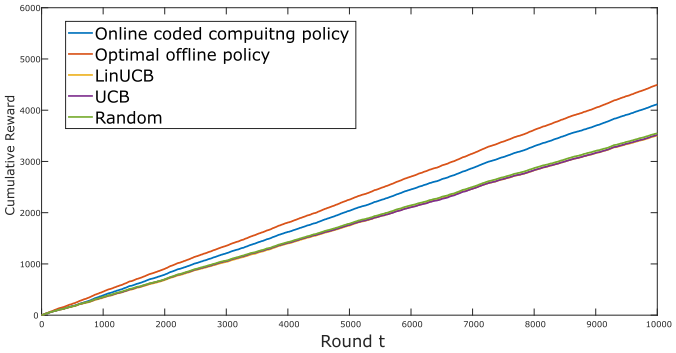

Fig. 5. Numerical evaluations for cumulative reward for Scenario 4.

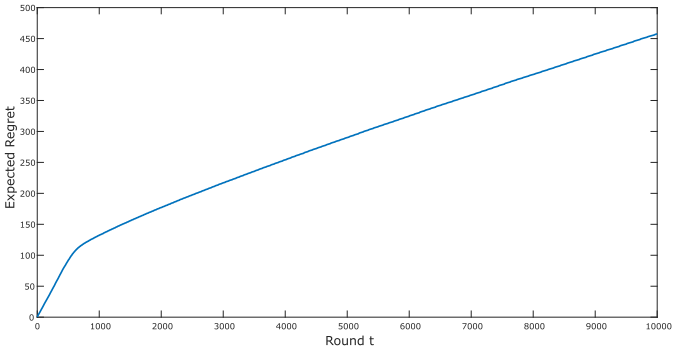

Fig. 6. Expected regret of the online coded edge computing policy for Scenario 1.

- The proposed online coded edge computing policy significantly outperforms other benchmarks by taking into account the context of edge computing network.
- Random and UCB algorithms are not effective since they do not take the context into account for the decisions. Although LinUCB is a contextual-aware algorithm, it achieves similar cumulative regret as random and UCB algorithms. That is because the success probability model is more general here than the linear functions that LinUCB is tailored for.

Fig. 6 presents the expected regret of the proposed policy for Scenario 1. We can conclude that the proposed policy achieves a sublinear regret in the time horizon T demonstrates the asymptotic optimality, i.e., $\lim_{T\to\infty} \frac{R(T)}{T} = 0$.

VI. Concluding Remarks and Future Directions

Motivated by the volatility of edge devices' computing capabilities and the quality of service, and increasing demand for timely event-driven computations, we consider the problem of online computation offloading over unknown edge cloud networks without the knowledge of edge devices' capabilities. Under the coded computing framework, we formulate a combinatorial-contextual multiarmed bandit (CC-MAB) problem, which aims to maximize the cumulative expected reward. We propose the online coded edge computing policy which provably achieves asymptotically-optimal performance in terms of timely throughput, since the regret loss for the proposed CC-MAB problem compared with the optimal offline policy is sublinear. Finally, we show that the proposed online coded edge computing policy significantly improves the cumulative reward compared to the other benchmarks via numerical studies.

Acknowledgment

The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Coded computing in unknown environment via online learning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 185–190.
[2] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments," in *Proc. Osdi*, vol. 8, 2008, p. 7.
[3] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones," in *Proc. NSDI*, vol. 13, 2013, pp. 185–198.
[4] Q. Yu, S. Li, N. Raviv, S. M. Mousavi, M. Soltanolkotabi, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," in *Proc. Artif. Intell. Statist.*, 2019, pp. 1215–1225.
[5] A. Eryilmaz, R. Srikant, and J. R. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 411–424, Apr. 2005.
[6] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
[7] J. G. Dai and W. Lin, "Maximum pressure policies in stochastic processing networks," *Oper. Res.*, vol. 53, no. 2, pp. 197–218, Apr. 2005.
[8] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.
[9] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 702–710.
[10] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Communication-aware scheduling of serial tasks for dispersed computing," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1330–1343, Aug. 2019.
[11] M. Hoseinnejhad and N. J. Navimipour, "Deadline constrained task scheduling in the cloud computing using a discrete firefly algorithm," *Int. J. Next-Gener. Comput.*, vol. 8, no. 3, pp. 198–209, 2017.
[12] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.
[13] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.
[14] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2100–2108.
[15] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2418–2422.

[16] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4403–4413.
[17] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3368–3376.
[18] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 34–40, Apr. 2017.
[19] S. Li *et al.*, "Coded computing," *Found. Trends Commun. Inf. Theory*, vol. 17, no. 1, pp. 1–148, 2020.
[20] S. Prakash, A. Reisizadeh, R. Pedarsani, and A. S. Avestimehr, "Coded computing for distributed graph analytics," *IEEE Trans. Inf. Theory*, vol. 66, no. 10, pp. 6534–6554, Oct. 2020.
[21] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1920–1933, Mar. 2020.
[22] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.
[23] N. Ferdinand and S. C. Draper, "Hierarchical coded computation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1620–1624.
[24] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "Draco: Byzantine-resilient distributed training via redundant gradients," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 903–912.
[25] C.-S. Yang and A. S. Avestimehr, "Coded computing for secure Boolean computations," *IEEE J. Sel. Areas Inf. Theory*, early access, Jan. 29, 2021, doi: 10.1109/JSAIT.2021.3055341.
[26] J. So, B. Guler, A. S. Avestimehr, and P. Mohassel, "CodedPrivateML: A fast and privacy-preserving framework for distributed machine learning," 2019, *arXiv:1902.00641*. [Online]. Available: http://arxiv.org/abs/1902.00641
[27] J. So, B. Guler, and A. S. Avestimehr, "A scalable approach for privacy-preserving collaborative machine learning," 2020, *arXiv:2011.01963*. [Online]. Available: http://arxiv.org/abs/2011.01963
[28] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler mitigation in distributed optimization through data encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5434–5442.
[29] J. So, B. Guler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *IEEE J. Sel. Areas Inf. Theory*, early access, Jan. 26, 2021, doi: 10.1109/JSAIT.2021.3054610.
[30] S. Prakash *et al.*, "Coded computing for low-latency federated learning over wireless edge networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 233–250, Jan. 2021.
[31] S. Prakash, A. Reisizadeh, R. Pedarsani, and A. S. Avestimehr, "Hierarchical coded gradient aggregation for learning at the edge," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 2616–2621.
[32] M. Yu, S. Sahraei, S. Li, S. Avestimehr, S. Kannan, and P. Viswanath, "Coded merkle tree: Solving data availability attacks in blockchains," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2020, pp. 114–134.
[33] S. Li, M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan, and P. Viswanath, "PolyShard: Coded sharding achieves linearly scaling efficiency and security simultaneously," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 249–261, 2021.
[34] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Timely-throughput optimal coded computing over cloud networks," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2019, pp. 301–310.
[35] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Timely coded computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 2798–2802.
[36] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, Mar. 1985.
[37] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.
[38] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, 2010, pp. 661–670.
[39] R. Sen, K. Shanmugam, M. Kocaoglu, A. Dimakis, and S. Shakkottai, "Contextual bandits with latent confounders: An NMF approach," in *Proc. Artif. Intell. Statist.*, 2017, pp. 518–527.
[40] R. Shariff and O. Sheffet, "Differentially private contextual linear bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4296–4306.
[41] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1466–1478, Oct. 2012.
[42] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1702–1710.
[43] L. Chen and J. Xu, "Task replication for vehicular cloud: Contextual combinatorial bandit with delayed feedback," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 748–756.
[44] S. Li, B. Wang, S. Zhang, and W. Chen, "Contextual combinatorial cascading bandits," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48, 2016, pp. 1245–1253.
[45] S. Muller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.
[46] L. Qin, S. Chen, and X. Zhu, "Contextual combinatorial bandit and its application on diversified online recommendation," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2014, pp. 461–469.

Chien-Sheng Yang (Graduate Student Member, IEEE) received the B.S. degree in electrical and computer engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2015. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Southern California (USC), Los Angeles. His interests include information theory, machine learning, and edge computing. He received the Annenberg Graduate Fellowship in 2016. He was a finalist of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) Best Paper Award in 2019.

Ramtin Pedarsani (Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Tehran, Tehran, Iran, in 2009, the M.Sc. degree in communication systems from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 2011, and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 2015. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA. His research interests include machine learning, optimization, information theory, game theory, and transportation systems.

Dr. Pedarsani was a recipient of the Best Paper Award at the IEEE International Conference on Communications in 2014, the NSF CRII Award in 2017, and the Communications Society and Information Theory Society Joint Paper Award in 2020.

A. Salman Avestimehr (Fellow, IEEE) received the B.S. degree in electrical engineering from the Sharif University of Technology in 2003, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California at Berkeley, Berkeley, in 2005 and 2008, respectively. He is currently a Professor, also the Inaugural Director of the USC-Amazon Center on Secure and Trusted Machine Learning (Trusted AI), and also the Director of the Information Theory and Machine Learning (vITAL) Research Lab, Electrical and Computer Engineering Department, University of Southern California. He is also an Amazon Scholar with Alexa AI. His research interests include information theory and coding theory, and large-scale distributed computing and machine learning, secure and private computing, and blockchain systems.

Dr. Avestimehr has received a number of awards for his research, including the James L. Massey Research and Teaching Award from the IEEE Information Theory Society, an Information Theory Society and Communication Society Joint Paper Award, a Presidential Early Career Award for Scientists and Engineers (PECASE) from the White House, a Young Investigator Program (YIP) Award from the U.S. Air Force Office of Scientific Research, a National Science Foundation CAREER Award, the David J. Sakrison Memorial Prize, and several best paper awards at conferences. He is currently a General Co-Chair of the 2020 International Symposium on Information Theory (ISIT). He has been an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY.