Opportunistic Coded Distributed Computing: An Evolutionary Game Approach

Yue Han*, Dusit Niyato†, Cyril Leung‡§, Dong In Kim¶,
* Alibaba-NTU Joint Research Institue, Nanyang Technological University, Singapore
†School of Computer Science and Engineering, Nanyang Technological University, Singapore
‡ Department of Electrical and Computer Engineering, The University of British Columbia, Canada
§ NTU-UBC Research Centre of Excellence in Active Living for the Elderly, Singapore.
¶ Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, South Korea
Email:hany0028@e.ntu.edu.sg, dniyato@ntu.edu.sg, cleung@ece.ubc.ca, dikim@skku.ac.kr

2021 International Wireless Communications and Mobile Computing (IWCMC) | 978-1-7281-8616-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/IWCMC51323.2021.9498947

***Abstract*—Task offloading has been proposed and studied to overcome the problem of energy and computation constrained terminals. Computationally intensive tasks are often parallelable, and therefore the execution time can be further improved via a coded distributed computing (CDC) approach, as CDC offers robustness against stragglers by introducing redundant computational tasks. In this paper, we study a user-centric task offloading problem, in which the edge performs the offloaded computation with CDC. Furthermore, the extent of the straggler's effect on servers is also unknown to the user. This requires users to explore server and code settings of the CDC, and "opportunistically" select the best combo to maximize the utility. For simplicity, we refer to this scenario as *opportunistic coded distributed computing*. We formulate the problem as an evolutionary game in which each user is self-interested. The payoff is calculated based on the monetary cost of CDC-as-a-Service and total delay, weighted by user-defined parameter values. For the game solution, an evolutionary stable equilibrium (ESS) is used, i.e., probabilistic joint selection of server and code configuration. To obtain the ESS, we present an iterative algorithm based on the revision protocol. A theoretical analysis of equilibrium in terms of existence,uniqueness, stationarity, and stability is provided. Numerical simulations are conducted to support the theoretical findings and the adaption of equilibrium states to the hyper-parameters.**

***Index Terms*—mobile edge computing, task offloading, computation offloading, coded distributed computing, evolutionary game theory**

I. Introduction

With the advent of 5G networks and the deployment of Artificial Intelligence (AI), more and more data-hungry, computation-intensive, AI-empowered applications will be realized at the edge of the network, e.g., the Internet of Things (IoT), Unmanned Aerial Vehicles (UAVs), and smart vehicles [1]. Those applications include but are not limited to augmented reality (AR), virtual reality (VR), online games, deep-learning-enabled applications such as object recognition, dynamic route maximization, speech translation, and context-aware recommendations. Such applications significantly increase the demand for computation at the edge. To overcome the problem of constrained energy and computation of end terminals, *task offloading*, i.e., offloading large amounts of computation from end-terminals to nearby edge servers, is a promising solution [2]. However, to meet the more stringent requirement for Quality-of-Service (QoS), advanced latency-reduction methods for offloaded tasks are still needed.

Parallelable tasks, such as matrix multiplication, data preprocessing, and gradient computing, are highly involved in computationally intensive tasks [3]. Therefore, *distributed computing* is a promising way to further reduce the computational delay when offloaded tasks are parallelable [4]. In distributed computing, the completion time of a task is determined by the slowest worker to send results back to the master. When worker nodes significantly delay sending results back to the master, they become so-called *stragglers*. The straggler effect is known to be a critical challenge in distributed computing, more prevalent on cheaper clusters [5].

To mitigate the straggler problem, *coded distributed computing (CDC)* has recently received a lot of interest [3], [6]. CDC has been known for its robustness against stragglers by introducing redundancy in sub-tasks via coding techniques, e.g., erasure code [4]. In CDC, a master node recovers the result from a subset of worker nodes, instead of waiting for all workers nodes to complete. Thus, the performance of distributed computing can be further improved by CDC. This inspired us to improve the offloading solutions by taking the advantage of the CDC. However, the extent of redundancy against stragglers in CDC is dependent on the *code configuration*, which could be optimized for offloaded tasks.

In this paper, we study a user-centric task offloading problem, in which users have large parallelable tasks to compute, and the edge servers offer CDC-as-a-Service. The code configuration is user-defined, reflecting users' different preferences for the robustness against stragglers in different offloaded tasks. As the extent of the stragglers effect and robustness

This research is supported by the National Research Foundation, Singapore under its Emerging Areas Research Projects (EARP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. Also, this research is supported, in part, by Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Singapore Joint Research Institute (JRI), National Research Foundation, Singapore, under AI Singapore Programme (AISG Award No: AISG-GC-2019-003), Singapore Energy Market Authority (EMA), Energy Resilience, under Grant NRF2017EWT-EP003-041, WASP/NTU grant M4082187 (4080), Singapore Ministry of Education (MOE) Tier 1 (RG16/20), and National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant 2021R1A2C2007638

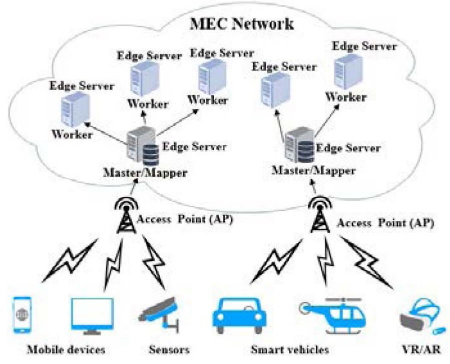


Figure 1. The proposed system model. MEC network consists of multiple edge servers. Edge servers that co-locate with access points are master nodes, connecting to other edge servers called worker nodes. Master nodes are responsible for collecting tasks from terminals, splitting the original tasks and encoding the sub-tasks according to the requested code configuration, and transferring sub-tasks to worker nodes, which perform the heavy computation.

brought by redundancy is unknown to users, this requires users to explore different combinations of server and code configurations, such that their expected utility is maximized. To reflect the uncertainty of this exploration phase, we refer to this scenario as *opportunistic coded distributed computing*.

We adopt a game-theoretical approach, a valuable framework for decentralized optimization, where players self-organize into mutually satisfying decisions in a competitive environment. To avoid the issue in traditional non-cooperative game theory, i.e., whether human beings can conform to the ideal of full rationality, we use a completely different approach, evolutionary game theory [7], in which successful strategies spread in the population, and less successful strategies wither, given reproductive fitness is derived from payoff. Such reproducing dynamics are similar to biological reproduction and do not require rational agents or other forms of cognitive abilities. Moreover, we take a stable evolutionary equilibrium (SEE) as the solution to the game. In summary, our main contributions to this paper are three-fold:

- We study a new user-centric task offloading problem, in which users have large parallelable tasks to compute and the edge servers provide CDC-as-a-Service. Using CDC, the total delay of offloaded tasks can be significantly reduced. As the effect of stragglers and code configuration is unknown, a user needs to explore server and code configurations jointly, and "opportunistically" select the best combo to maximize the utility.
- As the edge resources are finite, the problem is formulated as an "opportunistic coded distributed computing game" in the context of an evolutionary game. The solution to the game is a stable evolutionary equilibrium.
- We provide a theoretical analysis of the solution of the game, including uniqueness, existence, and stability with supporting evidence from extensive numerical results.

This paper is organized as follows. Section II reviews the related works. Section III introduces the system model and problem formulation. Section IV presents the evolutionary game formulation and theoretical analysis of the equilibrium. Section V presents the performance evaluation of our proposed game-theoretical decentralized optimization approach. Section VI summarizes the main findings.

II. Related Work

A. Task Offloading

Generally, how to optimally offload multiple tasks to multiple edge computing clouds is known to be an NP-hard combinatorial problem, various sub-optimal computational task offloading schemes have been proposed, e.g., binary (on-off) task offloading [2], [8], and partial task offloading [9], [10]. However, these approaches are not optimized for parallelable tasks, such as matrix-vector multiplication, widely used in data analytics algorithms. Those algorithms include principal component analysis (PCA), collaborative filtering (CF), logistic regression and convolutional neural network (CNN) [3].

B. Coded Distributed Computing (CDC)

CDC is a promising technique to tackle the straggler problem in distributed computing [4], [6], [11]. The study in [4] is the first to propose using erasure code (repetition code and MDS code) to speed up distributed computing. It considers in a homogeneous setup where workers and local data blocks are the same. In contrast, the study in [6] focuses on heterogeneous settings that allow different workers to receive different amounts of local data. A hierarchical coded computation scheme is proposed in [11] to make full use of the partial results of stragglers. However, these methods are based on a centralized approach and user-centric factors are not considered. Furthermore, the competition for limited edge resources is not considered. Furthermore, the competition of limited edge resources is not considered.

III. System Model

In this section, we introduce the system model of opportunistic coded distributed computing. We consider a set $I = \{1, 2, \dots, n\}$ of n users, and a set $P = \{1, 2, \dots, p\}$ of p access points (APs) where edge servers are co-located. Users can be further grouped into a set $G = \{1, 2, \dots, g\}$ of g populations, where users within a population are homogeneous, in terms of task request rate, task size, and sensitivity to delay. Computational tasks published by the users are assumed to be parallelable. To meet the stringent QoS requirements and overcome the computation limitations, users who want to offload large computations to edge servers, would further request a CDC paradigm to accelerate the performance. Edge servers in the MEC network are expected to offer CDC services for monetary gains. The code configurations of CDC are subject to user input. The Maximum Distance Separable (MDS) code [4] is used in this paper. The configuration of the MDS code is a tuple of positive integers, (n, k), where $n \geq k \geq 1$. The definition of CDC (with MDS code) is given in detail as follows:

Definition 1 (MDS-CDC)**.** CDC utilizing MDS code (n, k) includes the following steps:

1) *Encoding and Dispatching*: the original task is divided into k components of equal size by the master node. Then, an (n, k) MDS code is applied to each component, and a set $[n] := \{1, 2, \ldots, n\}$ of n encoded sub-tasks are obtained and then transferred to n worker nodes.
2) *Local Computation*: n nodes perform the computations with local computation functions $\langle f^i_{\mathbf{A}_i}(\cdot)\rangle_{i=1}^n$ and local encoded data blocks $\langle \mathbf{A}_i\rangle_{i=1}^n$. Upon completion of the local computing, the worker node sends the result back to the master node.
3) *Decoding*: a decoding function $dec(\cdot)$ is applied to a *minimal decodable set* $\mathcal{I}$ of workers to correctly recover the original results, where $\mathcal{I} \subset \mathcal{P}([n])$ and $\mathcal{P}([n])$ is the power set of $[n]$. In a CDC utilizing MDS code (n, k) , the decodable set $\mathcal{I}$ is any k worker nodes.

Note that the code configuration (n, k) collectively affect MDS-CDC performance. A larger k leads to a smaller local data block $\mathbf{A}_i$, less local computing time, but a large decodable set $\mathcal{I}$ and more stragglers effects.

Let $Q_p = \{(n_1, k_1), \ldots, (n_{q_p}, k_{q_p})\}$ be a set of q_p code configurations available in the AP $p \in \mathcal{P}$. A user can explore different access points and their available code configurations to maximize her utility. For the sake of convenience, we refer to the joint selection of the AP and code configuration (n, k, p) as a ***service***. Let $Q = \bigcup_{p\in\mathcal{P}} Q_p = \{s = (n, k, p)|p \in P, (n, k) \in Q_p\}$ be the set of all services edge servers can provide. All services are assumed approachable for a user.

A. Expected Run-time of an MDS-CDC task

In this section, we provide a run-time model of a CDC task with MDS code (n, k). According to the Definition 1, its run-time is k-th response from n nodes. Therefore, we first introduce a run-time model of a *single* node and then extend the model to the *k-th* response.

Probabilistic Run-time of a Single Node. Let T_i denote the stochastic run-time (in seconds) of the working node i, and assumes that the run-times of n nodes $T_1, T_2, \ldots, T_n$ are statistically independent. Previous works (e.g., [4], [6]) proposed T_i follows a shifted exponential distribution and the correctness of the proposition was verified by empirical studies on Amazon EC2 clusters in [4]. The cumulative distribution function (CDF) of T_i in [6] is:

$$\Pr[T_i \le t] = 1 - e^{-\frac{\mu_i}{l_i}(t - L_i\ell_i)}, \quad (1)$$

where μ_i is the straggler parameter of node i representing any reason for delay, e.g., communication, system failure, computation, and l_i is the number of row vectors allocated to the node i, a measure of the size of the local data block $\mathbf{A}_i$ given in the Definition 1. To make the scenarios simpler, we consider the scenario where helper workers utilized by a service s have same straggler parameters value, μ_s. Let D/k the data size allocated to each node, where D is the original data size. Thus, the CDF of a single node utilized by the service s $F(t|s)$ can be modified from (1) as follows:

$$F(t|s) := F(t|s; D, \mu_s, L) = 1 - e^{-\frac{\mu_s}{D/k}(t - L\frac{D}{k})}. \quad (2)$$

Probabilistic Run-time of the k-th Node. Next we give the run-time model of k-th response from n nodes. Given the workers of a service is homogeneous, k-th response is equivalent to a k-th order statistic as follows:

$$f_{(k)}(t|s) = nf(t|s)\binom{n-1}{k-1}F(t|s)^{k-1}(1-F(t|s))^{n-k}, \quad (3)$$

where $f(t|s)$ is the probability density function (pdf) of $F(t|s)$ in eq. (2). The mean run-time for a service s is:

$$\begin{aligned} E_1(s; D, \mu_s, L) &= \mathbb{E}_{T_{\text{coded}|s}\sim f_{(k)}}[T_{\text{coded}}|s] \\ &= \int_{t=L*D/K}^{\infty} nf(t|s)\binom{n-1}{k-1}F(t|s)^{k-1} \\ &\qquad (1 - F(t|s))^{n-k}dt. \quad (4) \end{aligned}$$

B. Queuing Delay

According to definition 1, data sent to the edge with a CDC request must be encoded and distributed to multiple workers before the actual computation. We assume that different CDC services can perform the above steps independently and simultaneously; however, tasks under the same CDC services need to form the queue because the edge resources are limited. We assume this is an $M/M/1$ queuing model, e.g., for transferring data to remote computing units in the edge computing environment. Therefore, the mean sojourn time is:

$$E_s = \frac{\delta}{\theta_s - \tilde{\lambda}_s} = \frac{\delta}{\theta_s - \sum_{u\in I}\lambda_u * \mathbb{I}\{d(u) = s\}}, \quad (5)$$

where δ denotes the equivalent unit time in seconds, θ_s denotes the rate of encoding and distribution of a particular CDC service s, $\tilde{\lambda}_s$ denotes the aggregated task arrivals rate, and $d(\cdot) : I \to Q$ denotes a mapping from a user to her selected services $s \in Q$, i.e., $d(u) = s$. Therefore, the total delay of service s requested by the user u is:

$$E(s; u) = E_s + E_1(s; D, \mu_s, L) \quad (6)$$

C. Total Cost

Assume that the resource is utilized at a cost, and the price charged per service $s \in Q$ is a constant, denoted by ψ_s. Similar to previous work [8], two objectives are combined into a single term, total cost C, via weighted sum:

$$C(s; u) := \alpha_u E(s; u) + (1 - \alpha_u)\psi_s \quad (7)$$

where α_u is a user-based hyper-parameter representing the sensitivity of monetary cost over latency. Thus,the problem becomes an integer programming:

$$\begin{aligned} \min_s \quad & \sum_{u\in I} C_u(s; u) \\ \text{s.t.} \quad & \tilde{\lambda}_s < \theta_s, \quad \forall s \in \mathcal{Q} \end{aligned} \quad (8)$$

Optimal solutions to eq. (8) can be achieved by enumeration, which requires high computational complexity. To reduce complexity, we propose a decentralized solution, namely an evolutionary game approach, described in the section below.

IV. Evolutionary Game Theory (EGT)

In this section, we first propose a formulation of the opportunistic distributed computing game. Second, an evolutionary stable strategy (ESS) is provided based on replicator dynamics. We then provide a theoretical analysis of the ESS and describe the implementation of the algorithm.

A. Game Formulation

In EGT, each player is programmed to play a strategy, and can be grouped into population based on the same properties. Population states, representing the distribution of strategies in the population, are identical to the mixed-strategy in static-payoff game. A opportunistic coded distributed computing game $G = (I, \Theta, f)$, defined by a set of players I, a strategy space Θ, and a combined payoff function f as follows:

- **players and population**: equivalent to the set of users I ($|I|= n$) and the set of population G ($|G|$=g) defined in the system model. Let n_j denote the size of a population $j \in G$, then we have $\sum_{j=1}^{g} n_j = n$.
- **pure strategy**: equivalent to the set of services Q defined in the system model. For notational convenience, the set is re-labeled as $Q = \{1, 2, \dots, m\}$.
- **population states**: is the vector $x_i = (x_{i1}, \dots, x_{im}) \in \Delta, \forall i \in G$, where x_{ik} is the portion of players selecting the CDC service k in population i, and $\Delta = \{x_i \in \mathbb{R}_+^m : \sum_{k=1}^{m} x_{ik} = 1\}$ is the *simplex* composed of all possible population states in a single population. Let $\Theta = \Delta^g$ be the **mixed-strategy space** of g population, and $x := (x_1, x_2, \dots, x_g) \in \Theta$ be the *mixed-strategy profile*.
- **payoff**: equivalent to the negative of total cost in eq. (7). Let $f_{ik} : \Theta \to R$ be the payoff function for a strategy $k \in Q$ in population $i \in G$, then $f_{ik}(x) = -C(k; i)$, i.e.,

$$f_{ik}(x) = -\alpha_i \left(E_1(k) + \frac{\delta}{\theta_k - \sum_{i \in G} \lambda_i n_i x_{ik}} \right) - (1 - \alpha_i)\psi_k. \quad (9)$$

  Let $f_i(\Theta) = (f_{i1}(\Theta), \dots, f_{im}(\Theta))$ be the collective payoffs from population i, and $f(\Theta) = (f_1(\Theta), \dots, f_g(\Theta))$ the payoffs vectors of the multi-population.

B. Revision Protocol

In the context of EGT, the probability of selecting a strategy for a user can be adapted through learning from other players. The adaption process, reflected by the evolution of population states, can be modeled by a framework called the *revision protocols*, which captures the conditional switch rate from one strategy to another. The most popular protocol is *replicator dynamics / pairwise proportional imitation protocol* leading to a system of Ordinary Differential Equations (ODEs):

$$\dot{x}_{ik} := \eta x_{ik}(f_{ik}(x) - \bar{f}_i(x)) \qquad \forall i \in G, k \in Q, \qquad (10)$$

where $\bar{f}_i(x) := \sum_{k \in Q} f_{ik}(x) x_{ik}$ is the *average payoff* for the population i, and $\eta \geq 0$ is the learning rate. The right-hand side of eq. (10) defines the associated vector field $\phi : R^{g \times m} \to R^{g \times m}$ for each population i, where $\phi_{ik}(x) = x_{ik}(f_{ik}(x) - \bar{f}_i(x))$. Let $T = (0, +\infty)$ denote the domain of evolutionary time, and $\xi(\cdot, x^0) : T \to \Theta$ be the (local) solution to eq. (10) such that $\frac{\mathrm{d}\xi(t,x^0)}{\mathrm{d}t} = \phi[\xi(t, x^0)]$.

Proposition 1. The right-hand side of eq. (10) defines the associated vector field $\phi : R^{g\times m} \to R^{g \times m}$ for each population i, where $\phi_{ik}(x) = x_{ik}(f_{ik}(x) - \bar{f}_i(x))$. Vector fields ϕ of the replicator dynamics eq. (10) satisfy the Lipschitz condition.

Proof. A brief proof is provided due to the limited space. To prove the Lipschitz condition, we can instead show there exists some contestant M such that $\left|\frac{\partial \phi_{ik}}{\partial x_{jh}}\right| = \left|\frac{\partial x_{ik}}{\partial x_{jh}} + x_{ik}\left(\frac{\partial f_{ik}(x)}{\partial x_{jh}} - \frac{\partial \bar{f}_i(x)}{\partial x_{jh}}\right)\right| \leq M$, for all $x \in \Theta$. This can be shown by the boundedness of $\frac{\partial x_{ik}}{\partial x_{jh}}$, f_i and $\frac{\partial f_i(x)}{\partial x_{jh}}$ given as follows: (1) $\left|\frac{\partial x_{ik}}{\partial x_{jh}}\right| \leq 1$, with maximum value achieved only when $(i,k) = (j,h)$; (2) the variable component in f_{ik}, $\left|\frac{\delta}{\theta_k - \sum_{i\in G}\lambda_i n_i x_{ik}}\right|$ is less than $\left|\frac{\delta}{\theta_k}\right|$, thus f_{ik} together with its weighted sum f_i are bounded; (3) $\left|\frac{\partial f_{ik}(x)}{\partial x_{jh}}\right| = \left|\frac{\delta}{(\theta_k - \sum_{i\in G}\lambda_i n_i x_{ik})^2}\right| \left|\sum_{i\in G}\lambda_i n_i \frac{\partial x_{ik}}{\partial x_{jh}}\right| \leq \left|\frac{\delta}{\theta_k}\right| \left|\sum_{i\in G}\lambda_i n_i\right|$, which is a constant independent of x. □

Theorem 1. A *unique* solution $\xi(\cdot, x^0)$ *exists* for every initial state $x^0 \in \Theta$ to the system of ODEs eq. (10).

Proof. Based on proposition 1, the vector field ϕ is Lipschitz continuous on Θ. By the Picard-Lindelöf theorem, the above conclusion is derived directly. □

After proving the solution $\xi(\cdot, x^0)$ exists for any initial state x^0 in eq. (10), we proceed to identifying the solution to the opportunistic coded distributed computing game. i.e., a set of *stable* and *stationary* states $x \in \Theta$ in eq. (10). We start with the the stationary states.

*a) **Stationarity**:* Stationary states Θ^{ST} is a set of states $x \in \Theta$ such that if the initial point $x^0 \in \Theta^{ST}$, then we have $\xi(t, x^0) = x^0$ for $t \geq 0$. Based on the replicator dynamics in eq. (10), the stationary states can be identified by letting

$$\dot{x}_{ik} = 0 \quad \forall i \in G, k \in Q. \qquad (11)$$

More specifically, eq. (11) is achieved via either $x_{ik} = 0$ or $f_{ik}(x) = \bar{f}_i(x)$. The former condition leads to a set of *boundary stationary points* (i.e., pure strategy lies on the vertex of the polyhedron Θ), and the latter leads to a set of *interior stationary points*, i.e., $x_{ik} \neq 0$ and $f_{ik}(x) = \bar{f}_i(x)$. This leads to the discussion of stability of states, i.e., whether the selection probabilities of CDC service at the equilibrium stage are robust to small disturbance, e.g., mutant strategy.

*b) **Stability**:* Two terms relate to the concept of stability: The most basic concept is Lyapunov stability, frequently referred to as stability for short. Intuitively, a state x is *Lyapunov stable* if no small perturbation of the state induces a movement away from x. Another term is *asymptotically stable* if the state is Lyapunov stable and all sufficiently small perturbations of the state induce a movement back toward x (reflected in the vector field), i.e., there exist a neighborhood B^* such that

$\lim_{t\to\infty} \xi(t, x^0) = x$ for all $x^0 \in B^* \cap \Theta$ [7]. It is not hard to verify that the boundary stationary points are not stable. Therefore, we only consider interior stationary points.

Theorem 2. The interior stationary points identified in eq. (11) are asymptotically stable.

Proof. Based on the Lyapunov direct method, we need to prove the time derivative of the Lyapunov function is strictly negative. Let $x^* = (x^*_{ik})_{i\in G, k\in Q}$ denote the interior evolutionary equilibrium and $e_{ik}(t) = x^*_{ik} - x_{ik}(t)$ be the error function over time t. Define the Lyapunov function $V_{ik} : T \to R_+$ with $V_{ik}(t) = \frac{e^2_{ik}(t)}{2}$, then the time derivative of V_{ij} is:

$$\dot{V_{ij}} = e_{ik}\dot{e_{ik}} = -e_{ik}\dot{x_{ik}} = -e_{ik}(t)\eta x_{ik}(f_{ik}(x) - \bar{f}_i(x)). \quad (12)$$

When $f_{ik}(x) - \bar{f}_i(x) > 0$, the population ratio x_{ik} will increases, hence $e_{ik}(t) = x^*_{ik} - x_{ik}(t) > 0$. As $x_{ik} \neq 0$, we have $\dot{V_{ij}} < 0$. Same results can be derived when $f_{ik}(x) - \bar{f}_i(x) < 0$. Based on the Lyapunov stability theory, the interior stationary point is asymptotically stable. □

Theretofore, $\dot{x}_{ik} = 0$ leads to a set of stable and stationary point, at which the strategy selections of different population converge to a set of interior evolutionary equilibrium, i.e. ESS.

C. Implementation

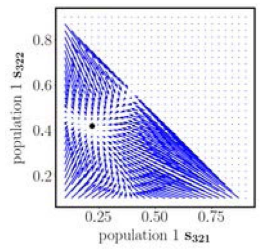
The replicator dynamics in eq. (10) can be implemented via an iterative algorithm, as shown in Algorithm 1. In this algorithm, we assume that the average payoff of each population is available to each of the users in the same population. The algorithm is a nested loop, and hence the complexity is $O(mg)$, where $g = |G|, m = |Q|$.

Algorithm 1: Joint AP and Code Configuration Selection Following the Replicator Dynamics

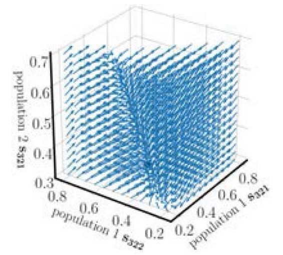
```
**Initialization**: $\forall i \in I$, randomly assign a strategy
  $k \in Q$; a small positive learning rate $\eta$; convergent
  threshold $\epsilon$; MAX_COUNT;
$t \leftarrow 1$;
**while** $|x_{ij}(t) - x_{ij}(t-1)| \geq \epsilon$ *and* $t \leq$ *MAX_COUNT*
 **do**
    **for** $i \in G$ **do**
        $k \leftarrow \text{Rand}(1, m)$ Randomly selects a service
          $k \in Q$;
        Calculate the payoff $f_{ik}$ according to (9);
        After the player receives information on payoff
          from all players in the same group, calculate
          the average payoff $\bar{f}_i$ ;
        Determine whether to swich to other strategy $h$
          according to the below revision protocol (10);
    **end**
    Update $t \leftarrow t + 1$
**end**
**Result:** Population States $x$ at the equilibrium stage;
          convergent payoffs $f$
```

Table I
SIMULATION PARAMETERS

Service	n	k	AP	μ	ψ	θ
s_{321}	3	2	1	1	1	90
s_{322}	3	2	2	1.2	1	100
s_{422}	4	2	2	1.2	1	90
Population	No. of users	α	λ	strategies		
1	80	0.8	1	$\{s_{321}, s_{322}, s_{421}\}$		
2	60	0.5	0.5	$\{s_{321}, s_{322}\}$		
Dynamics	δ	η				
	600 seconds	0.01				



(a) Case 1



(b) Case 2

Figure 2. Vector Field of the System and ESS

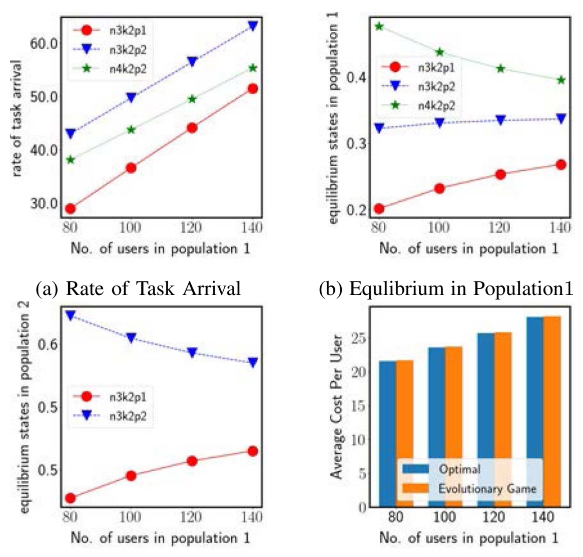
V. PERFORMANCE EVALUATION

In this section, we describe numerical simulations to evaluate the performance of service selections under different scenarios in the opportunistic coded distributed computing game. For ease of presentation, we consider an edge computing environment with two APs (i.e., $p = 2$) and two population of users (i.e., $g = 2$); however, the model is general and can be extended without significant overhead incurred. Assume the master co-located with AP 2 can approach to workers with higher computational capability, reflected by a larger straggler parameter μ [6]. A higher latency-sensitivity is assumed for population 1. Table I summarizes the parameter values for population, service and replicator dynamics.

A. Equilibrium Stability

We first analyze the dynamic behavior of populations using vector fields (VF). The VF ϕ specifies the direction and velocity of change of the population states, at each point x in the state space Θ. To visualize the result, we consider two scenarios. Case 1: only population 1 is considered in the game. Case 2: both population 1 and 2 are included in the game.

Note that population states in each population are subject to $\sum_{k\in Q} x_{ik} = 1, \forall i \in G$, so only first $|Q|-1$ states need to be analyzed, and the last state of each population can be inferred accordingly. As shown in the Fig. 2, VFs of both scenarios are clearly Lipschitz continuous, so the systems has a unique solution through every population state in the state space, and any unstable strategy will follow arrows in the VF to reach the ESS (marked by the black dot), i.e. asymptotically stable.



(a) Rate of Task Arrival (b) Equlibrium in Population1

(c) Equlibrium in Population2 (d) Average Cost

Figure 3. Equilibrium Adaptation

B. Adaption of Evolutionary Equilibrium

Then we analyze evolutionary equilibrium adaptation in Case 2 when certain system parameters change. Specifically, variation in the number of user in a population is studied. We fix n_2, the size of population 2 and vary n_1, the size of the population 1 from 80 to 140.

Fig. 3(a) shows that congestion status of each service, indicated by the task arrival rate, increases linearly with n_1. This is to be expected, since more users appear in the game, more tasks are expected to offloaded to edge servers. Fig. 3(b) shows that when n_1 is small, the portion of service s_{422} in ESS is larger in population 1. As users in population 1 are highly sensitive to delays, s_{422} can provides the fastest computation at the cost of more redundant blocks $(n-k)$. However, the increase of n_1 leads to the congestion of s_{422}, offsetting the benefit of latency reduction brought by the coding techniques. Therefore, when n_1 increases, the proportion of s_{422} in ESS decreases, while that of s_{322} and s_{321} increases.

Fig. 3(c) shows that changes in the number of users in population 1 also affect the service selection at equilibrium stage in population 2. This is due to s_{321} and s_{322} are shared edge resources. Population 2 is a group of users who are not sensitive to latency, and hence are not interested in exploring various code configurations. Their code configuration of interest is assumed to be $(n = 3, k = 2)$ (Table I). Both APs provide such a code configuration, and the users in population 2 need to explore which AP to offload the task to. As a higher computational capability is assumed for edge servers connected to AP 2, users in population 2 prefer s_{322} when it is not very congested. However, when more users in population 1 adapt their strategy to s_{322}, it becomes congested, and the benefit brought by the higher computational capability decreases. This leads to fewer users selecting AP 2.

Fig. 3(d) compares the proposed method and the baseline method in terms of average cost per user. Baseline is based on optimal decision profiles obtained by brute-force search, i.e., enumerating all possible decision profiles and finding the profiles that lead to the minimum total cost while satisfying the constraints in (8). The average costs obtained by the two methods are similar (Fig. 3(d)). Given the integer constraint of decisions, the optimal solution yields a slightly higher average cost than the proposed method. Compared to optimal solutions, decentralized solutions are more efficient with significantly less complexity and can be implemented more easily when the number of access points and code configuration increases.

VI. Conclusion

We proposed a new method, *opportunistic coded distributed computing* in this paper to address the delay issue in the user-centric task offloading problem. Evolutionary game was used to model the dynamic process of users' strategies with bounded rationality. We formulated the user utility based on the monetary cost of CDC-as-a-Service and total delay. We implemented the iterative algorithms based on replicator dynamics and performed numerical evaluations to obtain a game solution. These results confirmed the stability of the evolutionary equilibrium and its adaptation under various hyper-parameter values. The proposed method can be extended to larger decision profiles with low complexity. More complex scenarios, e.g., with only partial payoff information or collaboration among edge servers, can be studied for future works.

References

[1] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated Learning in Mobile Edge Networks: A Comprehensive Survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[2] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[3] K. T. Kim, C. Joe-Wong, and M. Chiang, "Coded Edge Computing," in *IEEE INFOCOM 2020*, Jul. 2020, pp. 237–246.

[4] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding Up Distributed Machine Learning Using Codes," *IEEE Transactions on Information Theory*, pp. 1514–1529, 2018.

[5] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, "Large Scale Distributed Deep Networks," *Advances in Neural Information Processing Systems*, 2012.

[6] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227–4242, 2019.

[7] J. W. Weibull, *Evolutionary game theory*. MIT press, 1997.

[8] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-Edge Computation Offloading for Ultradense IoT Networks," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4977–4988, 2018.

[9] Y. He, J. Ren, G. Yu, and Y. Cai, "D2d communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1750–1763, 2019.

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

[11] N. Ferdinand and S. C. Draper, "Hierarchical coded computation," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1620–1624.