Elastic Resource Allocation for Coded Distributed Computing over Heterogeneous Wireless Edge Networks

Journal:	*IEEE Transactions on Wireless Communications*
Manuscript ID	Paper-TW-Dec-21-1666.R1
Manuscript Type:	Original Transactions Paper
Date Submitted by the Author:	04-May-2022
Complete List of Authors:	Nguyen, Cong; University of Technology Sydney, School of Electrical and Data Engineering Nguyen, Diep; University of Technology Sydney Faculty of Engineering and Information Technology, Dinh , Hoang; University of Technology Sydney, School of Electrical and Data Engineering Phan, Tran Khoa; La Trobe University, Department of Computer Science and Information Technology Niyato, Dusit; Nanyang Technological University, School of Computer Engineering; Pham, Anh; Ho Chi Minh City University of Technology, Department of Computer Science and Engineering Dutkiewicz, Eryk; University of Technology Sydney Faculty of Engineering and Information Technology, Global Big Data Technologies Center
Keyword:	

Elastic Resource Allocation for Coded Distributed Computing over Heterogeneous Wireless Edge Networks

Cong T. Nguyen, Diep N. Nguyen, Dinh Thai Hoang, Khoa Tran Phan, Dusit Niyato, Hoang-Anh Pham and Eryk Dutkiewicz

Abstract

Coded distributed computing (CDC) has recently emerged to be a promising solution to address the straggling effects in conventional distributed computing systems. By assigning redundant workloads to the computing nodes, CDC can significantly enhance the performance of the whole system. However, since the core idea of CDC is to introduce redundancies to compensate for uncertainties, it may lead to a large amount of wasted energy at the edge nodes. It can be observed that the more redundant workload added, the less impact the straggling effects have on the system. However, at the same time, the more energy is needed to perform redundant tasks. In this work, we develop a novel framework, namely CERA, to elastically allocate computing resources for CDC processes. Particularly, CERA consists of two stages. In the first stage, we model a joint coding and node selection optimization problem to minimize the expected processing time for a CDC task. Since the problem is NP-hard, we propose a linearization approach and a hybrid algorithm to quickly obtain the optimal solutions. In the second stage, we develop a smart online approach based on Lyapunov optimization to dynamically turn off

Cong T. Nguyen, Diep N. Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz are with the School of Electrical and Data Engineering, University of Technology Sydney, NSW 2007, Australia (e-mail: cong.nguyen@student.uts.edu.au; diep.nguyen@uts.edu.au; hoang.dinh@uts.edu.au; eryk.dutkiewicz@uts.edu.au).

Cong T. Nguyen and Hoang-Anh Pham are with the Ho Chi Minh City University of Technology (HCMUT), Vietnam National University Ho Chi Minh City (VNU-HCM), Vietnam 700000 (e-mail: ntcong.sdh19@hcmut.edu.vn, anhpham@hcmut.edu.vn).

Khoa Tran Phan is with the La Trobe University, VIC 3086, Australia (e-mail: K.Phan@latrobe.edu.au).

Dusit Niyato is with the Nanyang Technological University, Singapore 639798 (e-mail:DNIYATO@ntu.edu.sg).

The conference version of this manuscript [1] is currently under review at IEEE WCNC 2022.

straggling nodes based on their actual performance. As a result, wasteful energy consumption can be significantly reduced with minimal impact on the total processing time. Simulations using real-world datasets have shown that our proposed approach can reduce the system's total processing time by more than 200% compared to that of the state-of-the-art approach, even when the nodes' actual performance is not known in advance. Moreover, the results have shown that CERA's online optimization stage can reduce the energy consumption by up to 37.14% without affecting the total processing time.

Index Terms

Coded distributed computing, Maximum Distance Separable code, resource allocation, INLP, Lyapunov optimization, edge computing, and straggling effects.

I. Introduction

The core idea of distributed computing is to distribute an intensive computing task among multiple nodes in a network, thereby utilizing the computing resources of multiple devices (e.g., edge and mobile devices). As a result, distributed computing has significant advantages over conventional centralized computing. First, distributed computing can achieve a high processing speed as the workload can be shared among various computing nodes. Second, distributed computing systems are scalable, i.e., more devices can be easily added, and thus low-cost devices can be utilized [11]. These outstanding advantages have enabled distributed computing to be widely applied in areas such as wireless edge computing [2], [3], Internet-of-Things [4], [5], and distributed learning [6], [7].

Despite its advantages, applications of distributed computing face critical challenges, especially in heterogeneous wireless edge networks. Particularly, straggling problems caused by uncertainties of computing processes, e.g., inconsistent computation time and/or failures, can lead to unpredictably severe latency in some computing nodes. The reason is that a task requires the calculated results from all nodes, and thus the total processing time of a task will be determined by the slowest node in the system. Moreover, communication links in wireless networks can be unstable, resulting in even more latency for the whole process [8]–[10]. Coupled with the straggling problems at the nodes, this can dramatically slow down the whole distributed computing system.

To address this straggling issue, Coded Distributed Computing [11] (CDC) has recently emerged to be an effective solution. The core idea of CDC is to assign redundant workloads to edge nodes via advanced coding theoretic techniques. Such redundancy can compensate for the uncertain computation and transmissions time, thereby improving the stability and latency of uncoded distributed computing [11]. Among CDC techniques, the maximum distance separable (MDS) code has been widely used [8], [11]. The general procedure of a CDC system using MDS code is illustrated in Fig. 1. First, once a task arrives, it can be encoded into n equal-sized sub-tasks and sent to n nodes. With the MDS code, any k results from those n sub-tasks can be used to derive the final result ($k \leq n$). Thus, the server only needs to wait for the fastest k nodes, thereby mitigating the effects of straggling edge nodes and unreliable wireless communication links.

It can be observed that there are three factors that have significant impacts on the effectiveness and efficiency of the CDC process using the MDS code. First, a higher value of k means that each node has a sub-task with a smaller size to process, as k is inversely proportional to each sub-task's size [11]. On the other hand, it also means that the server needs to wait for more nodes to send back their computing results to be able to complete the computing task. In cases where there are many straggling nodes and communication links, this can potentially lead to a very long processing time for the whole system. Second, besides the total processing time, the value of k dictates the efficiency and energy consumption of the CDC process. Specifically, since the final results can be obtained from the computations of any first k nodes, the computational workload of the remaining $n - k$ nodes will be wasteful. However, without such redundancy, the uncertainties of the nodes' performances might significantly prolong the processing time. Thus, it is critical to optimize the trade-off between the total processing time and energy efficiency in the CDC process. Third, most of the existing approaches, e.g., [14]–[17], [24], focus on optimizing the number of nodes (finding optimal values of (n, k)) without considering node selection. However, edge nodes often have different computing capacities and communication links in practice. Consequently, this can significantly impact the effectiveness of CDC in heterogeneous environments such as wireless edge networks. Therefore, to improve the future system performance for CDC processes over heterogeneous wireless edge networks, all three abovementioned factors must be carefully taken into account.

To address the above challenges, this work develops a novel optimization framework, namely CERA (Coded distributed computing Elastic Resource Allocation), to jointly minimize the total processing time and the energy consumption for the whole system. In particular, CERA consists of two stages. In the first stage, once a task arrives at the server, the server needs to make optimal decisions regarding the (n, k) values as well as which nodes to be selected. To this end, we first formulate the joint code and node selection problem as an Integer Non-linear Programming (INLP) optimization problem. Since this problem is NP-hard, we develop a linearization approach to transform the INLP into an equivalent Integer Linear Programming (ILP) problem. Moreover, we leverage the unique characteristic of this problem to develop a highly effective hybrid algorithm based on Branch-and-Bound (BB) and binary search algorithms. Both the linearization approach and the hybrid algorithm can help to solve the original NP-hard problem with a low complexity, allowing them to be deployed at the servers to efficiently obtain optimal solutions.

In the second stage of CERA, we observe that severely straggling computing nodes can be early/temporarily turned off (removed from their current task) to conserve the system's energy. For that, we design an online optimization approach based on Lyapunov optimization [30] to optimize the CDC system's control policy, i.e., removing straggling nodes from the task computing process. For example, considering a CDC system with $(n, k) = (5, 3)$, and at a certain time, we observe that 4 nodes have done more than 95% of their workload, while the last node has done only 10% of its workload due to straggling effects. In this case, we can turn off the straggling node because it is likely that it will not be among the first 3 fastest nodes to complete their task, thereby reducing wasteful energy for the whole system. Nevertheless, choosing which node to turn off is not an easy task because each node has a different energy consumption rate and removing off a node too early may negatively impact the system's total processing time. Therefore, we develop a dynamic optimization approach that helps the server to decide when and which nodes to turn off to save energy while ensuring the total processing time is not affected by the control decisions. Particularly, our Lyapunov-based approach can optimally control the trade-off between energy consumption and actual workload remaining at each node via an adjustable control parameter (which can be freely adjusted by the server). Moreover, this approach can minimize the redundancy in the CDC process without requiring perfect knowledge about the nodes' real straggling effects. Instead, our approach only relies on observable parameters such as expected straggling parameters and the actual remaining workload at the nodes. Furthermore, we theoretically prove that the average system's energy consumption can be effectively bounded by the control parameter. The major contributions of this paper can be summarized as follows:

- We propose a novel two-stage optimization framework, namely CERA, to jointly optimize the coding, node selection, and online control policy for CDC over a heterogeneous wireless edge network. To the best of our knowledge, this framework is the first in the literature that can simultaneously and dynamically minimize the processing time and energy consumption for the CDC.
- We develop an effective INLP model to jointly optimize the coding and node selection. Since this problem is nonlinear and NP-hard, we propose a linearization approach to transform the INLP model into an equivalent ILP problem. To further enhance the efficiency of CERA, we develop a novel hybrid algorithm that can significantly reduce the problem's computational complexity, enabling CERA to be employed at edge servers to quickly find optimal solutions.
- We develop a smart Lyapunov-based online optimization approach to dynamically optimize the control policy of the server. This approach enables the server to optimize the trade-off between the total processing time and the system's resource efficiency, thereby greatly improving resource utilization. We also theoretically prove that the average system's energy consumption can be effectively bounded by an adjustable control parameter.
- We conduct simulations based on real-world datasets to evaluate and compare the performance of our proposed approach with other existing approaches. The results show that our proposed approach can outperform the state-of-the-art approach [14] by more than 200% in the first stage. Moreover, the results also show that CERA can reduce the total energy consumption for the whole system by up to 37.14% without affecting the total processing time.

The rest of the paper is organized as follows. The related work is discussed in Section II. Section III presents the system model and the problem formulation. The CERA framework is presented in details in Section IV, and its performance is evaluated in Section V. Finally, conclusions are drawn in Section VI.

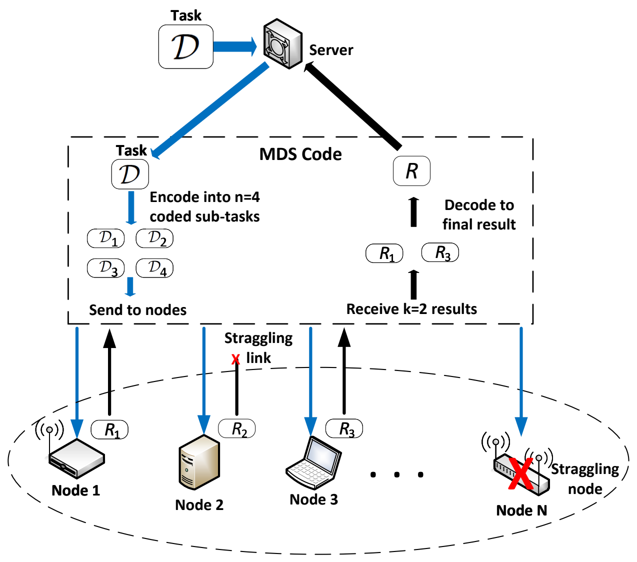

Fig. 1: Illustration of a coded distributed computing system. In this example, a task $\mathcal{D}$ is encoded into 4 sub-tasks $\mathcal{D}_1, \dots, \mathcal{D}_4$ using the $(n, k) = (4, 2)$ MDS code. The sub-tasks are then sent to 4 nodes (among a set of N nodes), namely Nodes 1, 2, 3, and N. Among them, Nodes 2 and N cannot send the results back due to straggling links and computation. Nevertheless, the server still can decode the final result based on the results of Nodes 1 and 3.

II. Related Work

In [14], the authors propose to apply the MDS code for matrix multiplication computations in a distributed learning system. The core idea of this approach is applying the MDS code to introduce redundancy in the computing process to compensate for the straggling effects. Moreover, the authors also find the optimal formulas for determining (n, k) and theoretically prove that the proposed solution can speed up the distributed learning process. Nevertheless, the formula and the results only apply for homogeneous settings, i.e., where all nodes have equal computing and straggling parameters. Similarly, in [15], the authors introduce a coding theoretic framework based on the MDS code to mitigate the straggling effects in distributed learning. Particularly, the authors propose to apply the MDS code to replicate portions of data across the nodes for gradient computing tasks. This replication ensures that the server only needs the results from a portion of the nodes to compute the final gradient. Simulation results show that the proposed framework can shorten the learning time by up to 2.4 times compared to that of the uncoded approach.

A similar framework is proposed in [16], where redundancy is introduced for datasets of linear regression tasks. This helps the server to compensate for the missing results from straggling nodes by using the redundant data. Simulation results show that the proposed framework can significantly improve the computing speed compared to that of uncoded distributed learning. However, similar to [14], the approaches in [15] and [16] only optimize the coding mechanism without considering the node selection and energy optimization for the whole system.

Moreover, the frameworks proposed in [14]–[16] do not take into account the straggling communication links. In wireless edge networks, these straggling links may have a significant impact on the total processing time, especially in the cases of distributed learning where large amounts of data need to be sent. To tackle this problem, a coded federated learning framework is proposed in [24]. In this framework, each node, besides its learning tasks, also sends a portion of its data to the server. The server can then use these data to compensate for the missing gradient updates caused by straggling. Moreover, the authors also propose to optimize the coding mechanism to find the optimal coding redundancy, taking into account the straggling parameters of the nodes and their communication links. Simulation results show that the framework can improve the total processing time by up to 15 times compared to that of uncoded approaches. Similar to [24], the coding mechanism proposed in [17] also considers the straggling effects of wireless communication links. However, unlike the abovementioned frameworks, this mechanism allows the nodes to submit partially computed results to the server, and the server can perform recovery computation on these results. In [33], an automatic repeat request scheme is proposed to address the straggling wireless links for computation over multiple access channels. Particularly, the authors propose new designs for transmitters and procedures for signaling to optimize the trade-off between rate and delay. In [18], a framework is proposed for designing optimal MDS coding schemes for CDC systems with packet erasure channels. Different from previous works that aim to minimize the expected total processing time, the authors in [18] introduce a metric *minimum latency*, i.e., a value of the total processing time that can be guaranteed with overwhelming probability, and focus on minimizing this metric. By thoroughly analyzing the considered system, the authors prove theoretical bounds of the latency and provide guidelines on designing optimal MDS codes depending on the channel conditions.

Another limitation of most existing works on CDC is that they do not take into account the different computational capacities of nodes. To address this gap, the authors in [20] propose a load allocation scheme for a CDC system where nodes with the same computing capability are grouped together. Moreover, the authors prove that their proposed scheme is optimal if there is a sufficiently large number of nodes in the system. Simulation results show that the proposed scheme can reduce the expected processing time by up to 52%. A similar framework is also proposed in [19]. However, unlike other works, this framework is developed specifically for optimizing the MapReduce scheme [21]. Simulation results show that the proposed framework can significantly reduce the communication load compared to state-of-the-art approaches. In [32], a new coding scheme is proposed to enable the computing nodes to further divide their assigned sub-tasks. Particularly, when a node receives its sub-task, the node can divide the sub-task into multiple blocks. Then, the node proceeds to compute each block sequentially, and the results of the blocks can be sent to the server separately. As a result, partial results from straggling nodes can be utilized. In [22], a new framework is proposed to design optimal load assignment scheme for distributed learning, which also takes into account the heterogeneity in computing capability of edge nodes. However, this work focuses more on the incentive aspect of distributed learning, and thus the coding scheme is optimized based on the nodes' computing costs.

Although all the abovementioned frameworks are shown to be effective in their respective systems, none of them can simultaneously address all the current challenges in CDC. Particularly, they either do not take into account the straggling effects of communication links or the heterogeneity of edge nodes. Recently, a new framework is introduced in [23] that can jointly optimize coding and node selection in coded distributed learning while taking into account the heterogeneity of edge nodes. To this end, the authors develop a deep reinforcement learning to find the optimal solutions to the joint optimization problem. Simulation results show that the proposed approach can reduce the total learning time of the system by up to 66%. Nevertheless, [23] and other existing frameworks cannot minimize the redundant energy consumption in CDC processes. To the best of our knowledge, our framework is the first in the literature that can dynamically control activities of edge nodes, thereby significantly reducing energy consumption with nearly no negative effects on the processing time of the whole system.

III. System Model

A. System Overview

Fig. 1 illustrates the CDC-based system that we consider in this paper. Moreover, the notations we use are summarized in Table I. The considered system consists of a server and a set $\mathcal{N}$ of N edge nodes. Using the (n, k) MDS code for a computing task $\mathcal{D}$, the server first divides the task into k sub-tasks with an equal size. For example, to multiply $\mathbf{A}\mathbf{x}$ and the $(3, 2)$ MDS code, the server can split $\mathbf{A} = [\mathbf{A}_1^T \mathbf{A}_2^T]^T$, and then assign 3 nodes to compute $\mathbf{A}_1\mathbf{x}$, $\mathbf{A}_2\mathbf{x}$, and $[\mathbf{A}_1 + \mathbf{A}_2]\mathbf{x}$, respectively. Then, these sub-tasks are encoded into n coded sub-tasks and sent to n nodes ($k \leq n \leq N$) [23], [24]. The nodes then perform the computations locally upon receiving their designated sub-tasks. As a node performs the computation, it can periodically measure the remaining workload and reports the percentage of work done to the server [13]. Once they finish, the nodes send the results back to the server. As soon as the server receives any first k results among those of the n sub-tasks, it can derive the final result of $\mathcal{D}$. For example, assume that the server decides to encode a task of size D using the $(n, k) = (4, 2)$ MDS code. As illustrated in Fig. 1, the server first splits the task into $k = 2$ sub-tasks of size $D_i = D/2$. Then, these sub-tasks are encoded into $n = 4$ coded sub-tasks ($\mathcal{D}_1$ to $\mathcal{D}_4$) using the MDS code. These 4 sub-tasks are then sent to 4 available nodes to compute. Then, for example, Node 1 and Node 3 are the first two nodes to return their results R_1 and R_2 to the server, respectively, while other nodes (e.g., Node 2 and Node 4) fail to compute/send the results due to straggling effects. With the MDS coding mechanism, the final result of $\mathcal{D}$ can be computed using only the results from Node 1 and Node 3.

In this case, the total time t_i needed for node i to complete a sub-task consists of the computation and the communication time, i.e., $t_i = t_i^s + t_i^c$, where t_i^s is the total time that the server needs to send sub-task $\mathcal{D}_i$ to node i and for node i to send result R_i back to the server, and t_i^c is the computation time required at node i to finish its assigned sub-task $\mathcal{D}_i$ [23], [24]. Let $\mathcal{K}$ be the set of the first k nodes that send the results back to the server. Then, the total actual processing time $\mathcal{T}$ of the CDC process can be defined by

$$\mathcal{T} = \max\{t_k : \forall k \in \mathcal{K}\}. \tag{1}$$

In the example shown in Fig. 1, assume that the completion time of each node (when it

TABLE I: Summary of Notations

Notation	Description
n	Number of nodes chosen using the MDS code
k	Number of fastest nodes among the n nodes
N	Total number of nodes in the system
$\mathcal{D}$	The computing task to be processed by the system
$\mathcal{D}_1 \ldots \mathcal{D}_i \ldots \mathcal{D}_n$	The n sub-tasks encoded using the (n, k) MDS code
t_i	Total time needed for node i to complete its assigned sub-task
t_i^s	The communication time of node i
t_i^c	The computation time of node i
$\mathcal{K}$	The set of the fastest k nodes
$\mathcal{T}$	The total actual processing time of $\mathcal{D}$
R_i	The result of sub-task $\mathcal{D}_i$
τ_i	The deterministic time for node i to upload/download a sub-task/result
N_i^d	Number of attempts required for node i to successfully download sub-task $\mathcal{D}_i$
N_i^u	Number of attempts required for node i to successfully upload its result R_i
p_i	The probability of the geometric distribution of N_i^d and N_i^u
t_i^d	The deterministic computation time of node i
t_i^r	The stochastic computation time of node i
η_i	The number of computations per second that node i can perform
α_i	The stochastic component of computation time from random memory access of node i
D	The size of task $\mathcal{D}$
ϕ	The total expected processing time of task $\mathcal{D}$
$c_i, x_i, y_i, \theta_i(t)$	Binary decision variables used in the optimization models
ρ	The maximum time required to solve a relaxed sub-problem
$t = 1, 2, \ldots, T$	Time is divided in T slots
$Q(t)$	The system's average workload at time slot t
$\mathcal{I}$	The set of n chosen nodes
$b(t)$	The average work done during slot t
ξ	The fixed duration of a time slot
$L(Q(t))$	The Lyapunov function
$\delta(Q)$	The Lyapunov drift
$E(t)$	The total energy spent during slot t
V	The control parameter

successfully sends the result back to the server) is $\{t_i\} = \{10, 90, 30, 150\}$. Then, $\mathcal{T}$ can be determined by the second smallest completion time, i.e., $\mathcal{T} = t_3 = 30$.

B. Communication Model

The communication time of a node depends on the communication link between the node and the server, which might be different across the nodes. Typically, the communication time can be determined by $t_i^s = \tau_i(N_i^d + N_i^u)$, where τ_i is the deterministic time to upload/download a sub-task. N_i^d and N_i^u are the numbers of attempts required for a successful transmission. We assume that N_i^d and N_i^u follow a geometric distribution with probability p_i. Moreover, we assume $N_i^d = N_i^u$ because the same channel can be used for uplink and downlink communication, and thus the retransmission probability is usually the same [8], [23], [24]. Then, the expected value of t_i^s is

$$\mathrm{E}[t_i^s] = 2\tau_i/p_i. \tag{2}$$

C. Computation Model

The computation time of a node consists of the deterministic and stochastic components, i.e., $t_i^c = t_i^d + t_i^r$. The deterministic computation time is given by $t_i^d = d/\eta_i = D/k\eta_i$, where η_i is the number of computations that node i can perform per second, and D is the size of (i.e., total number of computations required to complete) task $\mathcal{D}$ [23], [24]. The stochastic component of the computation time is assumed to follow an exponential distribution, i.e., $p_{t_i^r} = \lambda_i e^{-\lambda_i t}$, where $\lambda_i = \alpha_i\eta_i/d$, and α_i represents the stochastic component of computation time coming from random memory access [12], [23], [24]. Thus, the expected value of t_i^r is

$$\mathrm{E}[t_i^r] = 1/\lambda_i = D/k\eta_i\alpha_i. \tag{3}$$

IV. Two-stage Optimization Approach

To jointly minimize the processing time and energy consumption for the whole system, we develop a novel two-stage approach, namely CERA. In the first stage, the server solves the problem (**P1**) to obtain the optimal coding and node selection based on expected values of the straggling parameters. Then, in the second stage (during processing time), using our proposed Lyapunov-based optimization method, the server observes the performance of the wireless edge nodes and dynamically controls the nodes based on their current status to reduce the energy consumption. The two stages are presented in the following.

A. First Stage - Mixed Integer Programming

1) Problem Formulation: In the first stage, once a task arrives, the server needs to find the optimal values of the (n, k) code and select the best nodes to minimize the expected total processing time ϕ of a task, taking into account the heterogeneity of the wireless edge environment where the edge nodes may have different computing power η_i and transmission time τ_i. To this end, we formulate the considered problem as (**P1**) as follows:

$$(\mathbf{P1}) \min_{n,k,\mathbf{c},\mathbf{x}} \phi, \tag{4}$$

$$\text{s.t.} \sum_{i\in\mathcal{N}} c_i = n, \tag{5}$$

$$x_i \leq c_i, \qquad \forall i \in \mathcal{N}, \tag{6}$$

$$\sum_{i\in\mathcal{N}} x_i = k, \tag{7}$$

$$\phi \geq x_i\left(\frac{2\tau_i}{p_i} + \frac{D}{k\eta_i} + \frac{D}{k\eta_i\alpha_i}\right), \qquad \forall i \in \mathcal{N}, \tag{8}$$

$$k \leq n, \tag{9}$$

$$n \leq N, \tag{10}$$

$$c_i, x_i \in \{0, 1\}, \qquad \forall i \in \mathcal{N}, \tag{11}$$

$$n, k \in \mathbb{N}^+. \tag{12}$$

In (**P1**), the objective function (4) is the expected total processing time ϕ of task $\mathcal{D}$. Constraint (5) set the values for $\mathbf{c}$ which are binary variables representing the node selection decisions. Specifically, $c_i = 1$ if node i is among the n nodes selected, and there are exactly n nodes selected. Constraints (6) and (7) determine the values of $\mathbf{x}$ which are auxiliary variables introduced to determine the fastest k nodes. Particularly, $x_i = 1$ only when $i \in \mathcal{N}$ is among the fastest k nodes to send the results back to the server. Then, constraints (8) ensure that ϕ is bounded only by the k fastest nodes, i.e., if $x_i = 0$ then ϕ is not bounded by the completion time of node i.

The expected completion time of the node i is determined by

$$t_i^d + \mathrm{E}[t_i^r] + \mathrm{E}[t_i^s] = \frac{2\tau_i}{p_i} + \frac{D}{k\eta_i} + \frac{D}{k\eta_i\alpha_i}. \tag{13}$$

The remaining constraints set the conditions and value domains for n, k, c_i, and x_i.

2) Complexity of the INLP Formulation: From (**P1**), we can observe two characteristics regarding its complexity. First, (**P1**) is NP-hard as proven in Proposition 1.

Proposition 1. *The optimization problem* (**P1**) *is NP-hard.*

Proof: The considered problem can be decomposed into two sub-problems, namely MDS code optimization and node selection. The node selection sub-problem in (**P1**) is equivalent to the 0-1 knapsack problem [27]. Therefore, the node selection sub-problem is NP-hard. Consequently, the considered joint optimization problem is NP-hard. ■

Moreover, constraints (8) are nonlinear, and thus they make (**P1**) to be an INLP problem which is harder to solve than that of the ILP problem [28]. Thus, we first propose an effective linearization technique to transform (**P1**) into an equivalent ILP problem. As a result of the linearization, the problem can be effectively solved by using the BB algorithm. Given sufficient time, the BB algorithm can always find the optimal solutions for ILP models [29]. In contrast, optimal solutions are not guaranteed for all INLP models. Moreover, even if the optimal solution can be found, it requires much more time than that of ILP models.

3) Proposed Linearization Approach: To transform (**P1**) to an equivalent ILP problem, we first introduce the following constraints:

$$\sum_{j\in\mathcal{N}} j y_j = k, \tag{14}$$

$$\sum_{j\in\mathcal{N}} y_j = 1, \tag{15}$$

$$t_i + (1-x_i)M \geq \frac{2\tau_i}{p_i} + \sum_{j=1}^{N} y_j\left(\frac{D}{j\eta_i} + \frac{D}{j\eta_i\alpha_i}\right) \qquad ,\forall j \in \mathcal{N}, \tag{16}$$

$$\phi \geq t_i \qquad ,\forall j \in \mathcal{N}, \tag{17}$$

$$y_j \in \{0,1\}, \qquad \forall j \in \mathcal{N}. \tag{18}$$

Constraints (14) and (15) ensure that the newly introduced binary variables y_j equal 1 only when $j = k$. For example, if $k = 2$ then $y_2 = 1$, while $y_j = 0, \forall j \neq k$. Then, constraints (16) determine the expected processing time t_i of each node. In this constraint, M is a large number to ensure that t_i is bounded only when $x_i = 1$. Specifically, if $x_i = 0$, the left hand side of (16) becomes $t_i + M$. Since M is a large number, $t_i + M$ is always larger than the right hand side regardless of t_i, and thus the constraint is always satisfied. As a result, t_i is not bounded if $x_i = 0$. Conversely, when $x_i = 1$, the left hand side becomes t_i. Then, ϕ is bounded by the highest t_i in constraints (17). The ILP problem (**P2**) can now be defined by

$$(\textbf{P2}) \min_{n,k,\mathbf{c},\mathbf{x},\mathbf{y},\mathbf{t}} \phi, \tag{19}$$

$$\text{s.t. } (5)\text{-}(7),\ \ (9)\text{-}(18), \tag{20}$$

Next, we prove in Proposition 2 that (**P2**) is equivalent to (**P1**).

Proposition 2. (**P2**) *is equivalent to* (**P1**).

Proof: Let k' be the value of k in a feasible solution of the considered (**P1**). Then, (14) becomes $\sum_{j\in\mathcal{N}} j y_j = k'$. Moreover, since $\sum_{j=1}^{N} y_j = 1$ (from (15)), we have $y_{k'} = 1$ and $y_j = 0, \forall j \neq k^*$. Thus, (16) becomes:

$$\begin{aligned} t_i + (1 - x_i)M &\geq \frac{2\tau_i}{p_i} + \sum_{j\in\mathcal{N}} y_j \left(\frac{D}{j\eta_i} + \frac{D}{j\eta_i\alpha_i}\right) \\ &= \frac{2\tau_i}{p_i} + \left(\frac{D}{k'\eta_i} + \frac{D}{k'\eta_i\alpha_i}\right). \end{aligned} \tag{21}$$

Then, when $x_i = 1$, (17) becomes

$$\phi \geq t_i \geq \frac{2\tau_i}{p_i} + \left(\frac{D}{k'\eta_i} + \frac{D}{k'\eta_i\alpha_i}\right). \tag{22}$$

which is equal to (8). As a result, constraints (14)-(18) are equivalent to constraints (8), and the proof is now completed. ■

As a result of the proposed linearization, the BB algorithm can be employed to solve (**P2**) to find the optimal solution. However, the BB algorithm's computational complexity grows exponentially as the problem size increases. Particularly, the worst-case complexity of (**P2**) is $O(\rho 8^N)$, where ρ is the maximum time required to solve a relaxed sub-problem [29]. Therefore, in the following, by leveraging a unique characteristic of the considered optimization problem, we develop a hybrid solution to significantly reduce its computational complexity.

4) Hybrid Algorithm: Typically, a generic BB algorithm [29] can be applied to solve (**P2**) as follows. At the beginning (step 1), a relaxed master problem is created from (**P2**) by relaxing all integer restrictions on its decision variables, i.e., $\mathbf{c}$, $\mathbf{x}$, $\mathbf{y}$ n, and k. This relaxed problem forms the parent node. Then, using linear programming approaches [27], the problem is solved (step 2) to obtain the optimal solutions as well as the lower bounds of the objective value. Note that the obtained optimal solutions will most likely contain non-integer solutions, e.g., $0 < x_i < 1$. For each of these variables, two child nodes are then created, one with $x_i = 1$ and another one with $x_i = 0$. Next, the two relaxed sub-problems (step 3) are solved to obtain the solutions and bounds. At step 4, we choose one of the children nodes to branch. When a child node is solved, if there is no feasible solution found, the node can be eliminated (step 5). Otherwise, we continue to branch and create two more child nodes (step 6). Once an integer (incumbent) solution is found (step 7), the bounds of that solution and those of the other branches are compared. All the branches with worse bounds than those of the incumbent solutions can be removed from the search space (step 8), as they cannot produce better solutions. If an integer solution is found with better bounds than those of all the unexplored branches, it is the optimal solution, and the algorithm can be stopped.

In BB algorithms, the search space grows exponentially as the input size increases. Particularly, in the worst case where no branch can be eliminated, we have to explore all possible values of the binary variables. As a result, the worst-case complexity when applying the BB algorithm to solve (**P2**) is $O(\rho 2^{3N}) = O(\rho 8^N)$ because of the 3 types of decision variables $\mathbf{c}$, $\mathbf{x}$, and $\mathbf{y}$. Therefore, we propose a hybrid algorithm to significantly reduce the search space in the following.

First, we prove the monotonic behavior of the objective function over k in Theorem 1.

Theorem 1. *$\phi(k)$ is monotonically decreasing $\forall k < k^*$ and monotonically increasing $\forall k > k^*$, where k^* is the global minimum of $\phi(k)$.*

Proof: Proof of Theorem 1 is provided in Appendix A. ∎

An important result from Theorem 1 is that the monotonicity of ϕ over k on both sides of the

Algorithm 1 Proposed hybrid algorithm for (**P2**)

```
Input: Optimization problem (P2)
Output: Global optimal value $\phi^*$ and global optimal decisions $n^*, k^*, \mathbf{c}^*, \mathbf{x}^*, \mathbf{y}^*$
1:  $st \leftarrow 1$, $en \leftarrow N$, $mid \leftarrow (st + en)/2, \phi^* = 0$.
2:  repeat
3:      Branch on nodes $C_{mid-1}$, $C_{mid}$, $C_{mid+1}$ with $k = mid - 1$, $k = mid$, $k = mid + 1$ to
    find local optimum $\phi$ of each node.
4:      if $\phi(mid) < \phi(mid - 1)$ and $\phi(mid) < \phi(mid + 1)$ then
5:          $\phi^* \leftarrow \phi(mid)$, stop algorithm.
6:      else if $\phi(mid + 1) < \phi(mid - 1)$ then
7:          $st \leftarrow mid$, $mid \leftarrow (st + en)/2$
8:      else if $\phi(mid + 1) > \phi(mid - 1)$ then
9:          $en \leftarrow mid$, $mid \leftarrow (st + en)/2$
10:     end if
11: until $\phi^* > 0$
```

global optimum can be leveraged to reduce the search space of the BB algorithm, as presented in Algorithm 1. Generally, Algorithm 1 is a hybridization of binary search and BB algorithms. The general procedure of Algorithm 1 is as follows. First, we branch on the three nodes $C_{N/2-1}$, $C_{N/2}$ and $C_{N/2+1}$ to find their local optima. As proven in Theorem 1, if the middle node has the best objective value compared to those of the other two nodes, it is the global optimal solution. Otherwise, we compare the objective values of the remaining nodes to determine the search direction. Due to the monotonic behavior of $\phi(k)$ on both sides of the global optimum, we only need to search in the direction where $\phi(k)$ is decreasing over k. As a result, the other direction can be eliminated, thereby reducing one half of the remaining search space after each iteration. The algorithm continues until the optimal solution is found, i.e., until $\phi(k)$ of the middle node is lower than both those of the two adjacent nodes. With Algorithm 1, the computational complexity of the BB algorithm can be significantly reduced. Specifically, due to the hybridization with the binary search algorithm, Algorithm's 1 worst-case complexity is $O(\rho 4^N \log_2 N)$ (reduced from

2^N to $\log_2 N$ possible branches). Note that since only the branches that produce worse solutions are eliminated, the optimality of the global solution is guaranteed.

B. Second Stage - Online Lyapunov-based Optimization

The optimal solution in the first stage is calculated using the expected values of the straggling parameters of the wireless edge nodes. However, in practice, the straggling parameters often vary significantly from their expected values, resulting in unexpected performance. Moreover, since the core idea of CDC is to introduce redundancy to compensate for uncertainties, it is desirable to limit the redundant workload. Therefore, in the following, we propose a Lyapunov-based optimization approach that can minimize the redundancy in the CDC process without requiring complete knowledge about the nodes' real straggling effects. Our approach helps the server to decide when and which nodes to turn off to save energy with minimal impacts on the total processing time.

1) Lyapunov-based Approach: In our approach, time is divided into slots, denoted by $t = 1, 2, \dots, T$. Let $\mathcal{I}$ denote the set of the n nodes chosen in the first stage of CERA and $Q(t)$ denote the average workload remaining of those nodes at time slot t, i.e., $Q(t) = \frac{1}{n}\sum_{i\in\mathcal{I}} Q_i(t)$. We construct a virtual queue $Q(t)$, with initial value $Q(0) = \frac{1}{n}\sum_{i\in\mathcal{I}} Q_i(0) = \frac{1}{n}\sum_{i\in\mathcal{I}} \frac{D}{k}$. After each time slot, $Q(t)$ is updated by

$$Q(t+1) = \big(Q(t) - b(t)\big), \tag{23}$$

where $b(t) = \sum_{i\in\mathcal{I}} \xi \frac{\eta_i}{n(1+\alpha_i)}$ is the average work done during slot t, and ξ is the fixed duration of a time slot. This virtual queue represents the average remaining workload at the wireless edge nodes, and thus by minimizing the queue's backlogs we can minimize the total processing time. We define the Lyapunov function as $L(Q(t)) = \frac{1}{2}Q(t)^2$. Then, the Lyapunov drift can be defined as follow [30]:

$$\begin{aligned}\Delta(Q) &= \mathbb{E}\big(L(Q(t+1)) - L(Q(t))|Q(t)\big),\\ &= \frac{1}{2}\mathbb{E}\big[Q(t+1)^2 - Q(t)^2|Q(t)\big],\\ &= \frac{1}{2}\mathbb{E}\big[\big(Q(t)-b(t)\big)^2 - Q(t)^2|Q(t)\big],\\ &= \frac{1}{2}\mathbb{E}\big[b(t)^2 - 2b(t)Q(t)|Q(t)\big],\\ &\leq B - Q(t)\mathbb{E}\big[b(t)|Q(t)\big]\end{aligned} \tag{24}$$

where $B = \dfrac{1}{2}\big(\sum_{i\in\mathcal{I}}\eta_i(t)\xi\big)^2$. The drift $\Delta(Q)$ represents the changes in the queue's backlogs across each time slot. Next, the total energy spent during the time slot is defined by

$$E(t) = \sum_{i\in\mathcal{I}} E_i(t) = \sum_{i\in\mathcal{I}} \eta_i\xi, \tag{25}$$

where $E_i(t)$ is the energy consumed by node i during time slot t. Then, the drift-plus-penalty [30] can be defined by

$$\begin{aligned}&\Delta(Q(t)) + V\mathbb{E}\{\varphi(t)|Q(t)\}\\ =&\mathbb{E}\big[L(Q(t+1)) - L(Q(t))|Q(t)\big] + V\mathbb{E}\{\varphi(t)|Q(t)\},\\ \leq& B - Q(t)\mathbb{E}\big[b(t)|Q(t)\big] + V\mathbb{E}\{\varphi(t)|Q(t)\}.\end{aligned} \tag{26}$$

The drift-plus-penalty expression in (26) contains two objectives, i.e., minimizing the total processing time and reducing energy consumption. Moreover, the parameter V in (26) can be used to control the trade-off between the two objectives, e.g., a high value of V means a higher priority in saving energy. In this second stage, CERA aims to minimize the right-hand side of (26) for each time slot t, which is equivalent to solving the optimization problem (**P3**) as follows:

$$(\textbf{P3}) \quad \min_{\boldsymbol{\theta}(t)} \quad V\sum_{i\in\mathcal{I}}\theta_i(t)\xi - Q(t)\sum_{i\in\mathcal{I}}\frac{\theta_i(t)}{1+\alpha_i}\xi, \tag{27}$$

$$\text{s.t.} \quad \theta_i(t) \leq c_i, \tag{28}$$

$$\theta_i(t) \leq \theta_i(t-1), \tag{29}$$

$$\sum_{i\in\mathcal{I}}\theta_i(t) \geq k, \tag{30}$$

where $\boldsymbol{\theta}(t) = [\theta_1(t), \theta_2(t), \ldots, \theta_I(t)]$ are the binary variables representing the control decision of time slot t, i.e., $\theta_i(t) = 0$ means that node i is turned off in time slot t and vice versa. The objective of (**P3**) is to minimize both the energy consumption and the average remaining workload. To achieve this objective, the server decides to turn off some nodes in each time slot. These control decisions are constrained by (28)-(30). These constraints ensure that i) only the n nodes chosen at the beginning can be turned off, ii) once a node is turned off it cannot be employed for the task again[1], and iii) there must be at least k active nodes. Since the computational complexity of (**P3**) is low, i.e., there is only one decision variable vector $\boldsymbol{\theta}(t)$, it can be solved efficiently by standard BB algorithms to quickly obtain the control decisions for each time slot.

The main idea of this drift-plus-penalty optimization is to reduce the energy consumption based on the actual situation at the nodes. Since V is fixed throughout the time slots, at the beginning when the remaining workload is high, the server prioritizes reducing the workload. Therefore, few or no nodes will be turned off in the early time slots. As time progresses, the remaining workload will be reduced. As a result, the server will turn off more and more nodes later, thereby saving more energy. Moreover, the optimization problem (**P3**) is solved using only observable parameters, i.e., the expected values of the straggling parameters $\boldsymbol{\alpha}$ and the real remaining workload at each node. Thus, CERA does not require complete information on the nodes' performance in advance, which is intractable to obtain in practice.

The procedure of CERA can now be defined in Algorithm 2. Once a task $\mathcal{D}$ arrives, the server solves (**P2**) by using Algorithm 1 to obtain the optimal coding and node selection decisions. The server then encodes the tasks $\mathcal{D}$ using the optimal (n^*, k^*) values and sends them to the designated nodes (chosen using $\mathbf{c}^*, \mathbf{x}^*$). Next, at each time slot t, the server first observes the real-time values of $Q_i(t)$ to update $Q(t)$ accordingly. We assume that the time and energy it takes for the nodes to send the information regarding $Q_i(t)$ to the server are negligible. For example, sending a 2 kB message usually requires less than 5mJ and 2 ms [31]. Using these values, the

[1]Although employing a node for the same task again is possible, it is often usually impractical and inefficient in the context of CDC. This is because the slow nodes will be turned off first, and employing them again will add extra latency to their already slow computing processes. Moreover, this constraint helps to reduce the search space, as we only need to optimize the nodes that have not been turned off.

Algorithm 2 CERA

```
Input: $\boldsymbol{\eta}, \boldsymbol{\tau}, D, V, \xi$
Output: Optimal coding $(n^*, k^*)$, optimal node selection $\mathbf{c}^*, \mathbf{x}^*$, optimal control decision $\boldsymbol{\theta}^*$
1: Obtain $(n^*, k^*), \mathbf{c}^*, \mathbf{x}^*$ by solving (P2) using Algorithm 1.
2: for $t := 0$ to $T-1$ do
3:     Observe $Q(t)$
4:     Obtain $\boldsymbol{\theta}(t)$ by solving (P3)
5: end for
```

server solves (**P3**) to determine the control decisions for the time slot. This optimization process is repeated for each time slot until the stopping criterion is met, i.e., at least k nodes finish their workload.

2) Performance Analysis: We analyze and prove the bounds of the average workload and energy consumption in Theorem 2.

Theorem 2. *By applying CERA, the system 's average energy consumption satisfies*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[\varphi(t)|Q(t)\big] \leq \varphi^* + \frac{B}{V}, \tag{31}$$

where $\varphi^* = \sum_{i\in\mathcal{I}} \theta_i^* \eta_i \xi$ is the optimal energy consumption. Moreover, the average workload queue of the system satisfies

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[Q(t)] \leq \frac{V(\varphi^* - \varphi^{min}) + B}{\epsilon}, \tag{32}$$

where $\mathbb{E}[\varphi(t)] \geq \varphi^{min}, \forall t$ and $\mathbb{E}\big[b(t)|Q(t)\big] < \epsilon, \forall t$.

Proof: Proof of Theorem 2 is provided in Appendix B. ∎

An important result from Theorem 2 is that the system's average energy consumption can be effectively bounded based on the control parameter V. A higher value of V can lead to a lower energy consumption. However, it also increases the time-average workload at each node, which means that the total processing time might be longer. Choosing the value of V depends mostly on the server's priority, i.e., to save more energy or to prioritize reducing the total processing time. Nevertheless, as later shown in the performance evaluation, CERA can save energy without any negative impacts on the total processing time in many cases.

V. Performance Evaluation

A. Experimental Setup

1) CERA's first stage: We evaluate the performance of CERA's first stage in a CDC-based wireless edge network consisting of $N = 50$ nodes and 10 tasks with increasing task sizes (Task 1 has the smallest size). To this end, we use two datasets from closely related fields because there is currently no publicly available dataset for CDC. Specifically, we adopt the task sizes D from the Google Cloud Jobs dataset [25]. Moreover, we adopt the node capabilities parameters, such as τ_i and η_i, from the GWA-T traces dataset [26] which contains virtual machines' performance metrics. Furthermore, to clearly show the effects of task sizes on the optimal decisions, we arrange the tasks in ascending order of D. Finally, we set $p_i = 0.9$ and $\alpha_i = 2, \forall i \in \mathcal{N}$ [24]. The values of the parameters used in the simulations are summarized in Table II.

TABLE II: Values of Parameters Used in Simulations

Parameter	**Value**	**Parameter**	**Value**
N	50	D	[15000,900000]
τ_i	[0.05,30]	η_i	[1695,8932]
p_i	0.9	α_i	2

Moreover, we compare the performance of CERA's joint code and node selection optimization with the following baseline methods:

- *Myopic*: k is set to the maximum number of nodes, i.e., $k = N$. This approach is optimal if there is no straggling effects in the system.
- *OneNode*: In this approach, $k = 1$. This is equivalent to uncoded and centralized computing.
- *Static optimal code* [14]: k is determined by:
$$k = \left(1 + \frac{1}{W_{-1}(-e^{-\overline{\lambda}-1})}\right), \tag{33}$$
where $W_{-1}(.)$ is the lower branch of the Lambert W function and $\overline{\lambda}$ is the average straggling parameter of all nodes in the network.

Since these baseline methods do not optimize node selection, we use their k values and CERA's node selection optimization to determine the optimal decisions.

Then, to evaluate and compare the robustness of CERA's first stage optimization, we create 10 sets of parameters, e.g., computation and communication time, for 10 simulation runs. For each run, we use the decisions made by all the approaches without perfect information, i.e., using only the expected values, to calculate the real performance of the system with each set of parameters. Finally, average results of 10 runs are used to compare the performance of all approaches in each task.

2) CERA's second stage: Afterward, the performance of CERA's online optimization stage is evaluated. For ease of illustration, we first simulate a small CDC system with $(n, k) = (5, 3), D = 100, \boldsymbol{\eta} = [4, 6, 6, 6, 8], \xi = 2$, and $\boldsymbol{\alpha} = [2, 2, 2, 5, 10]$. We randomly generate the real values of $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ for each time slot using their corresponding distribution parameters. We then conduct the simulations with different values of V to evaluate how this control parameter can impact the performance of the system in terms of energy consumption and the amount of redundant workload reduced.

B. Simulation Results

1) CERA's first stage: Fig. 2(a) shows the expected total processing time of each task when using different approaches. As illustrated in the figure, our proposed approach outperforms all other approaches for all tasks. Particularly, our proposed approach can achieve ϕ that is up to 90 and 10 times lower than those of the *Myopic* and *OneNode* approaches, respectively. For the static optimal code approach, although it applies the optimal code in [14] and our proposed optimization to find the best nodes to perform, its performance is still not as good as that of our proposed framework. Specifically, for the largest-sized task, our proposed solution can reduce the expected total processing time up to 2.2 times compared with that of the static optimal code solution. The main reason is that, compared to our proposed approach, the static optimal code chooses a higher k. Although this reduces the sub-task size, the server has to wait for more nodes to send their results, and thus it may suffer more from the straggling nodes and communication links.

Fig. 2(b) shows the optimal values of k obtained by our approach and the static optimal code.

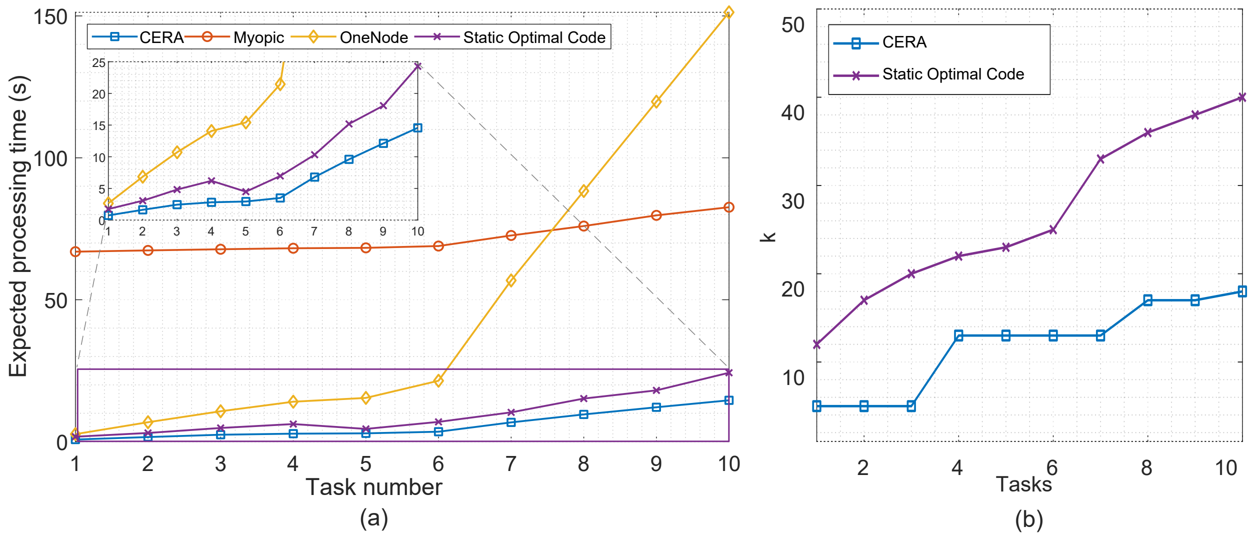

Fig. 2: Comparisons of the considered approaches in terms of (a) the expected total processing time and (b) the values of k.

Although their values are different, we can observe that as the task size increases, the value of k also increases. The reason is that for tasks with smaller sizes, the nodes' communication time has more impact on the expected total processing time. In this case, if k is high, the server has to wait for more nodes. In contrast, when the task size is larger, the nodes' communication time becomes insignificant compared to the computation time. As a result, a higher k is more desirable to reduce the workload at each node. However, if k is too high, the delay will be increased due to impacts of the straggling effects, as observed from the expected total processing time achieved by our proposed approach and the static optimal code approach in Fig. 2.

To further evaluate the performances of the approaches, we compare their average results in 10 runs (in terms of total processing time) with random parameters. For each approach, we use its optimal solution with imperfect information to see how the solution performs in cases of uncertainty. Fig. 3 shows the average total processing time of the considered approaches for each task. As observed from the figure, our proposed approach still outperforms the other approaches for all tasks. Moreover, as the task size increases, the proposed approach performs much better than the baseline methods. For example, for Task 10, our approach achieves a total processing time of 26.11, which is 2.4 times lower than that of the static optimal code approach.

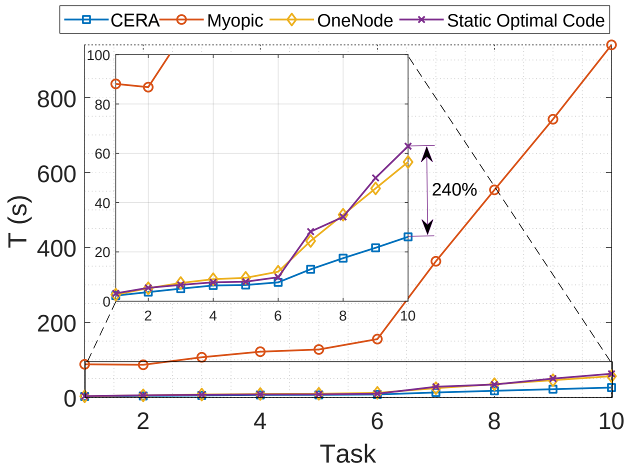

Fig. 3: Average total processing time of 10 runs.

Additionally, an interesting observation can be made regarding the *OneNode* and *Myopic* approaches. For the tasks with larger sizes, the *Myopic* approach performs much worse than its expected performance as can be seen from Fig. 2 and Fig. 3. Conversely, the *OneNode* approach performs much better than expected. The reason is that since we randomly generate the computation and communication time in this case, there are some values that will be much higher than their expected values, while there are also some values that will be much lower than the expected ones. For the *OneNode* approach, its performance depends only on the best node. Therefore, it can benefit from the unexpected good performance of the nodes. In contrast, the *Myopic* approach relies on the performance of every node, and the total processing time in this case is equal to the processing time of the worst node. Consequently, this approach suffers the highest straggling effects.

Furthermore, to clearly show the relation between the task size and the communication time, we examine the processing time of the nodes in the smallest-sized task (Task 1) and the largest-sized task (Task 10). Note that the optimal k for Task 1 is 5, whereas the optimal k for Task 10 is 18. As shown in Fig. 4, the communication time is a significant factor in executing Task 1. For example, communication constitutes up to 19.5% of the expected total processing time of node 3 at Task 1. In contrast, for Task 10, the communication time occupies only 1.1% of the processing time of the same node. Thus, for tasks with large sizes, a high k can significantly

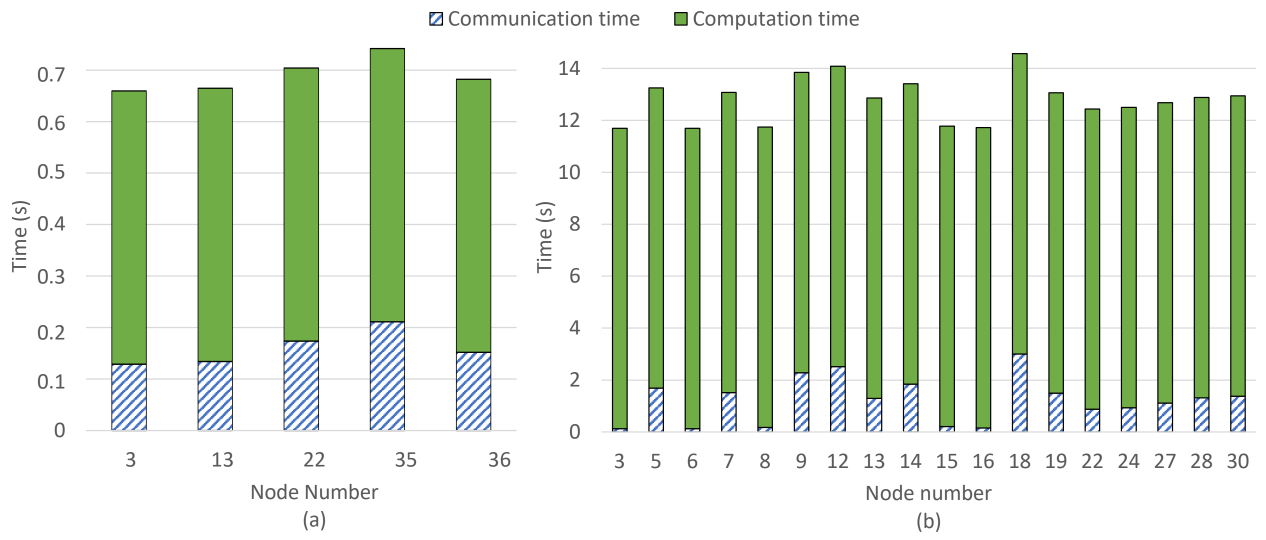

Fig. 4: Processing time components of (a) Task 1 and (b) Task 10.

reduce the computation time which has much higher impacts on the expected total processing time.

2) CERA's second stage: Fig. 5 illustrates the results of CERA's online optimization stage for a CDC system. As observed in Fig. 5(a), in the early time slots, no node in the system is turned off. As time progresses, the workload at each node decreases. Note that the amount of workload done by each node is different across time slots due to the random straggling effects which vary over time. At time slot 11, CERA decides to turn off Node 5. The reason is that Node 5 consumes the most energy per time slot among the nodes, even though its remaining workload is not much higher than that of Nodes 4 and 1. Afterward, Nodes 2 and 3 finish their sub-tasks at around time slot 17. Since $k = 3$, at this time, the server only needs one more result to finish the task. At time slot 19, CERA decides to turn off Node 4 because it still has more workload to perform and a higher energy consumption rate compared to those of Node 1. Finally, Node 1 finishes its sub-task at time slot 25, and the server can derive the final results from the work of Nodes 1, 2, and 3. Moreover, we simulate the case where CERA is not applied. As observed from the figure, even if Nodes 4 and 5 are not turned off, they will not contribute to the final result as they finish after Node 1.

Fig. 5(b) illustrates the energy consumption of each node in the above example. As observed

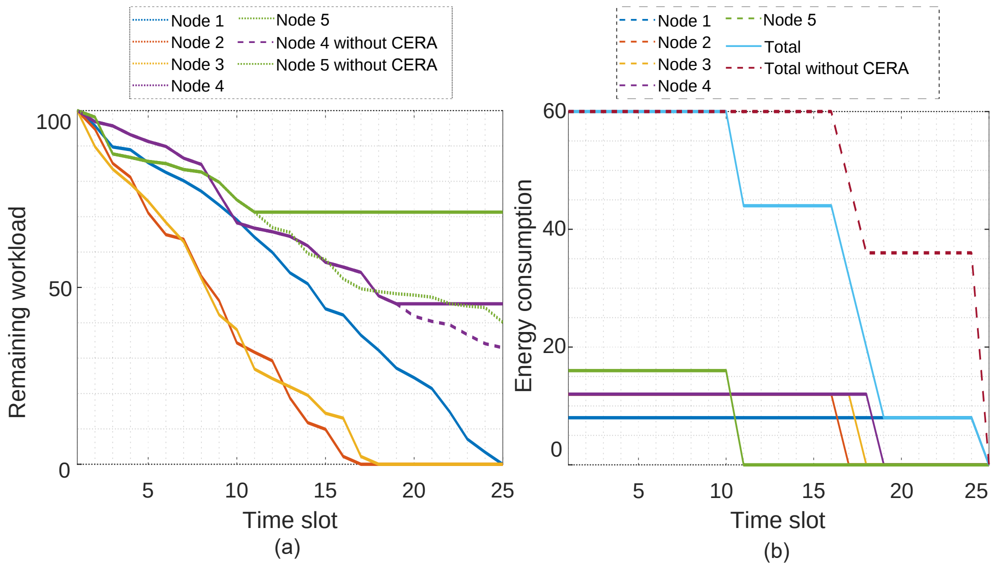

Fig. 5: Results of CERA online optimization for a CDC system with $V = 5$: (a) the workload remains of each node, and (b) the energy consumption of each node

from the figure, the energy consumption of each node in each time slot goes to zero as the node either is turned off or finishes its sub-task. Moreover, we show the total energy consumed by each node for each time slot, with and without CERA. The figure clearly shows that CERA can save a significant amount of energy by removing the nodes, thereby reducing the unnecessary redundant workload. This can save up to 23.49% of energy consumption in this example, without negatively impacting the total processing time. Next, we adjust the control parameter $V = 8$ while keeping all the remaining parameters the same to show the effect of V. As illustrated in Fig. 6, compared to the case when $V = 5$, Nodes 4 and 5 are turned off earlier in this case. Particularly, Nodes 4 and 5 are turned off at time slots 14 and 4, which are 5 and 7 slots earlier than the previous case, respectively. This can save up to 37.14% of energy consumption, an extra of 13.65% compared to that of the case where $V = 5$.

Nevertheless, a high value of V does not always produce the best result. We simulate another example similar to the previous one, except that $\alpha_1 = 2$ instead of $\alpha_1 = 1$. As illustrated in Fig. 7, CERA makes the same decisions as before, i.e., turn off Nodes 4 and 5. However, in this case, the simulated results without applying CERA show that if Node 4 is not turned off early, it can actually finish before Node 1, thereby improving the total processing time. The reason

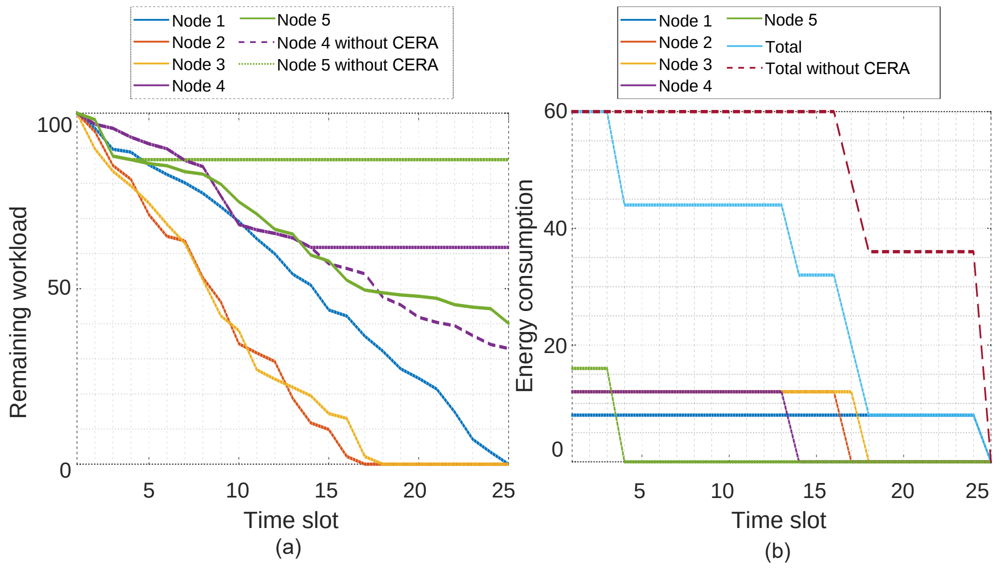

Fig. 6: Results of CERA online optimization for a CDC system with $V = 8$: (a) the workload remains of each node, and (b) the energy consumption of each node

is that we double the straggling parameter of Node 1 compared to previous cases, and thus its actual performance is much worse than expected. Nevertheless, we would like to emphasize that, the decision made by CERA is optimal according to the imperfect information available at the time. Particularly, at time slot 16 where Node 4 is turned off, its remaining workload is still higher than that of Node 1, and its energy consumption is also higher. Moreover, the actual performances of the nodes are intractable to be accurately predicted in practice. Furthermore, even if the total processing time is higher (by 5 slots), the percentage of energy saved in this case is remarkable, i.e., 41.99%. Given that this aligns with the server's priority in practice and the parameter V can be freely adjusted, CERA is a powerful tool for the server to optimize their CDC processes.

VI. Conclusion

In this work, we have developed CERA, a novel framework to jointly optimize the processing time and energy consumption for CDC processes over wireless edge networks. Particularly, we have developed a novel two-stage optimization framework. At the first stage, we have modeled a joint coding and node selection optimization problem to minimize the expected processing time

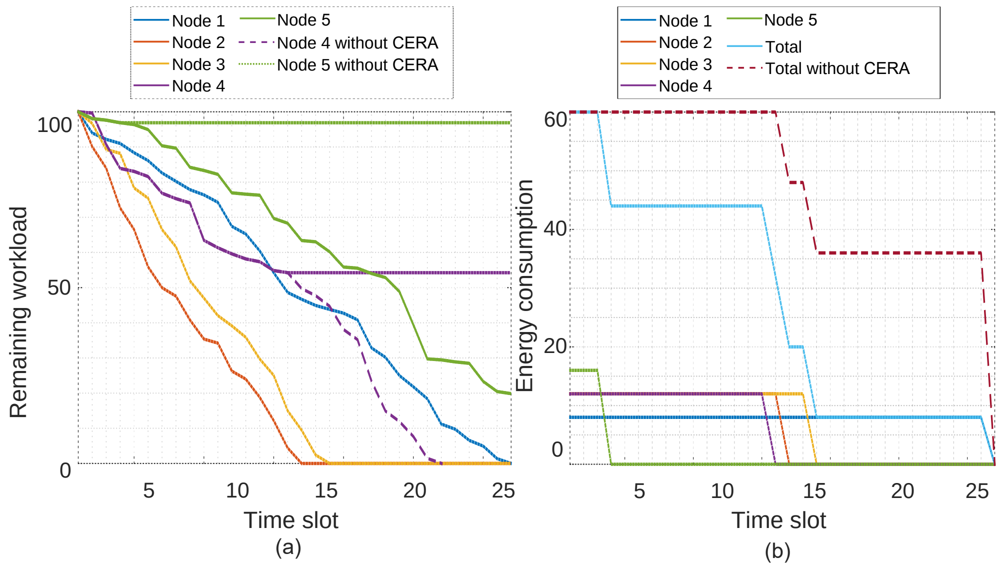

Fig. 7: Results of CERA online optimization for a CDC system with $V = 8$ and $\alpha_1 = 2$: (a) the workload remains of each node, and (b) the energy consumption of each node

for the system. Then, we have developed a linearization approach to reduce the computational complexity of the original problem. After that, a hybrid algorithm has been developed, which can solve the considered problem much more efficiently than the conventional BB algorithm while preserving the optimality of the obtained solutions. At the second stage, we have developed a Lyapunov-based dynamic optimization approach to optimize the online control policy for CDC processes. This approach can significantly reduce the energy consumption with minimum adverse impacts on the total processing time and without requiring perfect knowledge of the nodes' real performance. Simulations using real-world datasets have shown that our proposed approach can reduce the system's total processing time by more than 200% compared to that of the state-of-the-art approach, even with imperfect information of the node's capabilities. Moreover, the results have also shown that CERA's online optimization stage can reduce the energy consumption by up to 37.14% without affecting the total processing time. For future research directions, we can consider other coding schemes, e.g., [32], that allow asymmetric workloads for computing nodes, which might significantly improve the efficiency of CDC systems.

References

[1] C. T. Nguyen, D. N. Nguyen, D. T. Hoang, HA. Pham and E. Dutkiewicz, "Jointly optimize coding and node selection for distributed computing over wireless edge networks ," Jun. 2021, *arXiv preprint arXiv:2106.05475*. [Online]. Available: https://arxiv.org/abs/2106.05475

[2] T. Qiu et al, "Edge computing in industrial Internet of Things: Architecture, advances and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462-2488, July 2020.

[3] X. Hu, K. Wong and K. Yang, "Wireless Powered Cooperation-Assisted Mobile Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2375-2388, Apr. 2018.

[4] H. El-Sayed et al., "Edge of Things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706-1717, Dec. 2018.

[5] S. Xia, Z. Yao, Y. Li and S. Mao, "Online Distributed Offloading and Computing Resource Management With Energy Harvesting for Heterogeneous MEC-Enabled IoT," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6743-6757, Oct. 2021.

[6] W. Y. B. Lim et al., "Federated Learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031-2063, Apr. 2020.

[7] Y. Zhang, B. Di, Z. Zheng, J. Lin and L. Song, "Distributed Multi-Cloud Multi-Access Edge Computing by Multi-Agent Reinforcement Learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2565-2578, April 2021.

[8] D. Ko, S. H. Chae and W. Choi, "MDS Coded Task Offloading in Stochastic Wireless Edge Computing Networks," *IEEE Transactions on Wireless Communications*, Early Access, doi: 10.1109/TWC.2021.3109448.

[9] F. Wang, J. Xu and S. Cui, "Optimal Energy Allocation and Task Offloading Policy for Wireless Powered Mobile Edge Computing Systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2443-2459, April 2020.

[10] F. Wu and L. Chen, "Latency Optimization for Coded Computation Straggled by Wireless Transmission," *IEEE Wireless Communications Letters*, vol. 9, no. 7, pp. 1124-1128, July 2020.

[11] J. S. Ng et al., "A Comprehensive Survey on Coded Distributed Computing: Fundamentals, Challenges, and Networking Applications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1800-1837, thirdquarter 2021.

[12] J. Zhang and O. Simeone, "On Model Coding for Distributed Inference and Transmission in Mobile Edge Computing Systems," *IEEE Communications Letters*, vol. 23, no. 6, pp. 1065-1068, June 2019.

[13] M. A. Attia and R. Tandon, "Combating computational heterogeneity in large-scale distributed computing via work exchange," Nov. 2017, *arXiv preprint arXiv:1711.08452*. [Online]. Available: https://arxiv.org/abs/1711.08452

[14] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514-1529, Mar. 2018.

[15] R. Tandon, Q. Lei, A. G. Dimakis and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," *International Conference on Machine Learning*, Sydney, Australia, Aug. 06-11, 2017, pp. 3368-3376.

[16] C. Karakus, Y. Sun, S. N. Diggavi and W. Yin, "Redundancy techniques for straggler mitigation in distributed optimization and learning," *The Journal of Machine Learning Research*, vol. 20, no. 72, pp. 1-47, Apr. 2019.

[17] K. Taik Kim, C. Joe-Wong and M. Chiang, "Coded edge computing," *IEEE International Conference on Computer Communications (INFOCOM)*, Toronto, ON, Canada, 6-9 Jul. 2020, pp. 237-246.

[18] D. J. Han, J. Y. Sohn and J. Moon, "Coded wireless distributed computing with packet losses and retransmissions," *IEEE Transactions on Wireless Communications*, Early Access, doi: 10.1109/TWC.2021.3091465.

[19] F. Li, J. Chen and Z. Wang, "Wireless MapReduce Distributed Computing," *IEEE Transactions on Information Theory*, vol. 65, no. 10, pp. 6101-6114, Oct. 2019.

[20] F. Xu, S. Shao and M. Tao, "New Results on the Computation-Communication Tradeoff for Heterogeneous Coded Distributed Computing," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2254-2270, Apr. 2021.

[21] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters" *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, Jan. 2008.

[22] N. Ding, Z. Fang, L. Duan and J. Huang, "Optimal Incentive and Load Design for Distributed Coded Machine Learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2090-2104, July 2021.

[23] N. Van Huynh, D. T. Hoang, D. N. Nguyen and E. Dutkiewicz, "Joint Coding and Scheduling Optimization for Distributed Learning over Wireless Edge Networks," *IEEE Journal on Selected Areas in Communications*, Early Access, doi: 10.1109/JSAC.2021.3118432.

[24] S. Prakash et al., "Coded computing for low-latency federated learning over wireless edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 233-250, Jan. 2021.

[25] A. Hussain and M. Aleem, "GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures," *Data*, vol. 3, no. 4, pp. 38-50, Dec. 2018.

[26] S. Shen, V. Van Beek and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Shenzhen, China, 4-7 May 2015, pp. 465-474.

[27] M. X. Goemans, *Advanced Algorithms*.Cambridge, MA: MIT Laboratory for Computer Science, 1994.

[28] J. Kronqvist, D. E. Bernal, A. Lundell and I. E. Grossmann, "A review and comparison of solvers for convex MINLP" *Optimization and Engineering*. vol. 20, no. 2, 20(2):pp. 397-455, June, 2019.

[29] D. R. Morrison, S. H. Jacobson, J. J. Sauppe and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 1, no. 19, pp. 79-102, Feb. 2016.

[30] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol.3, no.1, pp. 1-211, Sep. 2010.

[31] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," *IEEE INFOCOM 2001*, Anchorage, AK, Apr. 22-26, 2001, pp. 1548-1557.

[32] L. Chen, K. Han, Y. Du and Z. Wang, "Block-Division-Based Wireless Coded Computation,"" *IEEE Wireless Communications Letters*, vol. 11, no. 2, pp. 283-287, Feb. 2022.

[33] L. Chen, N. Zhao, Y. Chen, F. R. Yu and G. Wei, "Toward Optimal Rate-Delay Tradeoff for Computation Over Multiple Access Channel," *IEEE Transactions on Communications*, vol. 69, no. 7, pp. 4335-4346, July 2021.

Appendix A

Proof of Theorem 1

We first relax the integer restriction of k for $t_i(k)$ and $\phi(k)$ to obtain the relaxed function $t_i(\kappa)$ and the relaxed objective function $\phi(\kappa)$, respectively. We now prove that $t_i(\kappa)$ is convex over κ by taking its second-order derivative:

$$\frac{\partial^2 t_i}{\partial \kappa^2} t_i(\kappa) = \frac{2D}{\kappa^3 \eta_i} + \frac{2D}{\kappa^3 \eta_i \alpha_i} > 0. \tag{34}$$

Therefore, $t_i(\kappa)$ is strictly convex over κ. Moreover, since $\phi(\kappa)$ is defined by the maximum of $t_i(\kappa)$, i.e., maximum of convex functions, $\phi(\kappa)$ is also strictly convex over $k(\kappa)$. Then, from the convexity of $\phi(\kappa)$, we have:

$$\phi(\varepsilon\kappa_1 + (1-\varepsilon)\kappa_2) < \varepsilon\phi(\kappa_1) + (1-\varepsilon)\phi(\kappa_2), \tag{35}$$

$\forall \kappa_1, \kappa_2, \forall 0 \le \varepsilon \le 1$. Now, without loss of generality, assume that we have $k-1 < k < k+1 < k^*$ where k^* is the global minimum of $\phi(k)$. Suppose for the sake of contradiction that $\phi(k-1) < \phi(k)$ and $\phi(k) > \phi(k+1)$. Adding both inequalities yields $\phi(k-1)+\phi(k+1) < 2\phi(k)$. However, from (35), if we choose $\kappa_1 = k-1$, $\kappa_2 = k+1$, and $\varepsilon = 0.5$, we have:

$$\begin{aligned} \phi(0.5(k-1) + 0.5(k+1)) &< 0.5\phi(k-1) + 0.5\phi(k+1), \\ 2\phi(k) &< \phi(k-1) + \phi(k+1), \end{aligned} \tag{36}$$

which contradicts the inequalities. Therefore, $\phi(k-1) < \phi(k) < \phi(k+1)$, and thus the function $\phi(k)$ monotonically decreases $\forall k < k^*$. The proof can be straightforwardly extended to prove that $\phi(k)$ monotonically increases $\forall k > k^*$.

Appendix B
Proof of Theorem 2

The proof of Theorem 2 is adapted from [30]. We have

$$\begin{aligned} \Delta(Q) &= \mathbb{E}\big(L(Q(t+1)) - L(Q(t)) | Q(t)\big), \\ &= \frac{1}{2}\mathbb{E}\big[Q(t+1)^2 - Q(t)^2 | Q(t)\big], \\ &= \frac{1}{2}\mathbb{E}\big[\big(Q(t) - b(t)\big)^2 - Q(t)^2 | Q(t)\big], \\ &= \frac{1}{2}\mathbb{E}\big[b(t)^2 - 2b(t)Q(t) | Q(t)\big], \\ &\le B - Q(t)\mathbb{E}\big[b(t)|Q(t)\big], \\ &\le B - \epsilon Q(t) \end{aligned} \tag{37}$$

where $B = \frac{1}{2}\big(\sum_{i\in\mathcal{I}} \eta_i(t)\xi\big)^2$ and $\epsilon > 0$ is a constant such that $\mathbb{E}\big[b(t)|Q(t)\big] < \epsilon$ for all slots.

For every slot, we have the drift-plus-penalty

$$\begin{aligned}&\Delta(Q(t)) + V\mathbb{E}\{\varphi(t)|Q(t)\}\\ =&\mathbb{E}\big[L(Q(t+1)) - L(Q(t))|Q(t)\big] + V\mathbb{E}\{\varphi(t)|Q(t)\},\\ \leq& B - \epsilon Q(t) + V\varphi^*,\end{aligned} \tag{38}$$

Using the law of iterated expectations [30], we have

$$\begin{aligned}\mathbb{E}\big[L(Q(t+1))\big] - \mathbb{E}\big[L(Q(t))\big] + V\mathbb{E}\{\varphi(t)\} \leq& B - \epsilon Q(t)\\ &+ V\varphi^*,\end{aligned} \tag{39}$$

Summing over $t = [0, 1, \ldots, T-1]$ and using the law of telescoping sum [30] yields

$$\begin{aligned}\mathbb{E}\big[L(Q(T-1))\big] - Q(0) + V\sum_{t=0}^{T-1}\mathbb{E}\{\varphi(t)\} \leq& T(B + V\varphi^*)\\ &- \epsilon\sum_{t=0}^{T-1}\mathbb{E}[Q(t)].\end{aligned} \tag{40}$$

Rearranging (40) and dividing by VT yields:

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{\varphi(t)\} \leq& \frac{-\mathbb{E}\big[L(Q(T-1))\big] + Q(0)}{VT} + \frac{V\varphi^* + B}{V}\\ &- \frac{\epsilon}{VT}\sum_{t=0}^{T-1}\mathbb{E}[Q(t)],\\ \leq& \frac{-\mathbb{E}\big[L(Q(T-1))\big] + Q(0)}{T} + \frac{V\varphi^* + B}{V}\end{aligned} \tag{41}$$

As $T \to \infty$, we have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\frac{-\mathbb{E}\big[L(Q(T-1))\big] + Q(0)}{T} = 0. \tag{42}$$

As a result, we have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\varphi(t)\big] \leq \varphi^* + \frac{B}{V}. \tag{43}$$

For the energy consumption bound, rearranging (40) and dividing it by ϵT yields:

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[Q(t)] \leq & -\frac{V}{\epsilon}\sum_{t=0}^{T-1}\mathbb{E}\{\varphi(t)|Q(t)\}\\ & + \frac{-\mathbb{E}\big[L(Q(T-1))\big]+Q(0)}{\epsilon T} + \frac{V\varphi^* + B}{\epsilon},\\ \leq & \frac{-\mathbb{E}\big[L(Q(T-1))\big]+Q(0)}{\epsilon T}\\ & + \frac{V(\varphi^* - \varphi^{min}) + B}{\epsilon}\end{aligned} \tag{44}$$

where φ^{min} is a constant such that $\mathbb{E}[\varphi(t)] \geq \varphi^{min}, \forall t$. As $T \to \infty$, we have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\frac{-\mathbb{E}\big[L(Q(T-1))\big]+Q(0)}{\epsilon T} = 0. \tag{45}$$

Therefore, we have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[Q(t)] \leq \frac{V(\varphi^* - \varphi^{min}) + B}{\epsilon}, \tag{46}$$

and thus the proof is completed.