Wireless Merged-r LT Coded Computation: A Low-Latency Design for Non-Linear Tasks

Borui Fang, Li Chen, *Senior Member, IEEE*, Yunfei Chen, *Senior Member, IEEE*, and Weidong Wang

***Abstract*—Coded computation has attracted significant attention because it can eliminate the stragglers' effect effectively. Most existing works of coded computation are designed for linear tasks, such as matrix multiplication. They cannot handle non-linear tasks directly, leading to high computation, transmission and decoding latency. This is not suitable for latency-sensitive services. In this paper, considering a non-linear task in wireless heterogeneous networks, we propose an efficient merged-r Luby transform (LT) coded computation scheme based on the rateless and sparse LT code. First, we give the merged-r LT coding strategy to reduce the computation and transmission costs. Then, the maximum degree decoding (MDD) strategy is proposed to speed up the decoding process. Finally, we analyze the latency performance for the whole network by designing the optimal merging parameter and sub-block size. The wireless non-linear merged-r LT coded computation (WNLMrLTCC) algorithm minimizes the total latency. Theoretical analysis and numerical simulation show that our proposed scheme has significant advantages over the existing ones for non-linear tasks.**

***Index Terms*—Coded computation, maximum degree decoding, merged-r LT coding, non-linear tasks, wireless heterogeneous networks.**

I. Introduction

In distributed computing networks [1], the latency performance for performing a large-scale computational task is bottlenecked by the slow nodes, also known as stragglers. To mitigate the stragglers' effect, a novel framework named coded computation is proposed by introducing the necessary computation redundancy to reduce the execution time, using only specific fast nodes.

The simplest coded computation was the repetition coding scheme [2], where computational tasks were replicated to be executed in multiple nodes. Since the duplicated tasks led to huge redundancy, the authors of [3] applied maximum distance separable (MDS) code to speed up the matrix-vector multiplication. It could provide a dramatic performance improvement in computation latency over the repetition coding scheme. The authors of [4] extended the MDS coding scheme to accelerate the matrix-matrix multiplication. For the heterogeneous networks with nodes of disparate computation capabilities, the authors of [5] provided an optimal computation load allocation for the MDS coding scheme to minimize the computation latency. Considering the transmission latency in wireless networks, the authors of [6] analyzed the total latency and designed the optimal MDS coding scheme with minimum latency. Furthermore, in order to handle not only computation stragglers but also transmission stragglers, the corresponding MDS coding schemes with the optimal trade-off between computation and transmission latency were studied in [7] and [8].

Compared to MDS code with a fixed rate and dense encoded symbols, Luby transform (LT) code is rateless and sparse, which makes it possible to make full use of the computed results from stragglers and reduce the coding overhead. Utilizing this rateless property, the authors of [9] proposed the LT coding scheme based on the sub-block division to exploit the partial results from all the nodes including stragglers. To further reduce the encoding and decoding complexity, the work in [10] presented a LT coding scheme with feedback information. Considering transmission overhead and transmission errors, respectively, the authors of [11] and [12] designed the corresponding LT coding schemes to balance both computation and transmission latency for wireless heterogeneous networks.

Coded computation has been studied for different computational requirements in [13]–[18]. For example, to reduce the computation latency for sparse matrix multiplication, the authors of [13] presented several coding schemes that imposed constraints on the extent to which coding was allowed. Based on the different sparsity levels for matrixes in different batches, the authors of [14] designed a novel coding scheme with the optimal task assignment. To remain information-theoretically private from nodes, the authors of [15] presented a rateless coding scheme for the private matrix multiplication. The work in [16] studied the problem of Byzantine attack identification in coded computation. In order to protect both the security and privacy of user data, the work in [17] designed the secure and private coded computation scheme. Moreover, the work in [18] presented the approximating coded distributed computing scheme when the exact computational result was not required.

Using the coding schemes discussed above, coded computation has been implemented in various complex distributed network scenarios. The authors of [19] considered a general MapReduce framework and characterized the optimal computation-communication tradeoff in coded distributed computing. For variability of computing speed in cloud networks, the authors of [20] designed an optimal coded computation load allocation strategy to maximize the timely

This work was supported by Natural Science Foundation of China (Grant No. 62471449), Industrial Technology Basic Project of MIIT (No. TC220A04M), and Anhui Provincial Natural Science Foundation (No. 2308085J24). An earlier version of this paper was presented in part at the 16th International Conference on Wireless Communications and Signal Processing (WCSP 2024), Hefei, China, 24–26 October 2024. *(Corresponding author: Li Chen.)*

Borui Fang, Li Chen, and Weidong Wang are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (e-mail: fangbr@mail.ustc.edu.cn; chenli87@ustc.edu.cn; wdwang@ustc.edu.cn).

Yunfei Chen is with the Department of Engineering, Durham University, DH1 3LE Durham, U.K. (e-mail: Yunfei.Chen@durham.ac.uk).

computation throughput. Considering a heterogeneous multi-hop network, the work in [21] proposed a hierarchical coding scheme to address the stragglers' effect caused by the slow worker and relay nodes. The authors of [22] designed a joint allocation algorithm for a multi-master heterogeneous-worker coded distributed computing scenario. As for a realistic hierarchical compute cluster, the work in [23] presented a fault-tolerant coded computation scheme to achieve the order-optimal computation. Furthermore, coded computation was also studied in more practical environments, including mobile edge computing [24], the Internet of Vehicles (IoV) [25], and satellite-terrestrial integrated networks [26].

All the above works consider linear tasks, such as matrix multiplication. But non-linear tasks are more prevalent in practice, especially in machine learning algorithms. To this end, several works on coded computation for non-linear tasks have been conducted. For the polynomial computing problem, the authors of [27] presented the Lagrange coding scheme and provided an optimal design for resiliency, security and privacy. Based on the Lagrange coded computation, the work in [28] developed a Lagrange coded blockchain model for private and consortium Internet of Things (IoT) systems. In order to solve the distributed gradient descent problem without the stragglers' effect, the authors of [29] proposed a gradient coding scheme with the prior knowledge of stragglers. Subsequently, the extensions of coded computation to federated learning and regression problems were studied in [30] and [31], respectively. Considering a linearly separable computation where the desired task could be expressed as a linear combination of numerous non-linear sub-tasks, the authors of [32] found the converse bounds for the optimal communication cost with the minimum computation cost. The authors of [33] further expanded on the above non-linear computational task in multi-user scenarios. Moreover, the works in [34] and [35] presented the corresponding rateless coding schemes to solve the non-linear computational tasks. However, these existing works for non-linear tasks are only simple extensions of those for linear tasks. They disregard the heterogeneity of non-linear computational tasks and nodes. Hence, for wireless heterogeneous networks, they cannot provide low latency. To design efficient coded computation schemes for non-linear tasks, several challenges need to be addressed.

1) **High Computation and Transmission Costs.** Because of the non-additivity for non-linear sub-tasks, the encoding pre-computation is precluded. This makes the magnitude of encoded sub-tasks much larger than that of linear tasks. Thus, the costs of computing and transmitting these encoded sub-tasks increase significantly. It is essential to reduce these costs to efficiently complete non-linear tasks.

2) **Inefficient Decoding Design.** In the existing decoding design, such as the belief propagation (BP) decoding strategy, each sub-task is precisely recovered. But for most machine learning algorithms, such as gradient descent, only the sum of these non-linear sub-tasks is of interest, making it unnecessary to decode the exact result of every sub-task.

3) **Heterogeneous Sub-Tasks Allocation.** Due to the heterogeneity of non-linear sub-tasks, the complexities of computing different sub-tasks may also be different. Thus, the existing computation load allocation strategies for linear computational tasks cannot handle non-linear tasks directly.

To address the above challenges, we propose a merged-r LT coded computation scheme with novel encoding and decoding strategies. Specifically, we first present the merged-r LT coding strategy based on the rateless and sparse LT code. Then, the maximum degree decoding (MDD) strategy is given to recover the sum of sub-tasks directly. Finally, through analyzing and minimizing the computation, transmission and decoding latency for the whole network, we obtain the optimal merging parameter and sub-block size. The main contributions of this paper are summarized as follows:

- **Merged-r LT Coding Strategy.** To reduce the computation cost of encoded sub-functions, we apply the rateless and sparse LT code to speed up non-linear tasks. Based on the idea of merging operations in the encoding process, the merged-r LT coding strategy is proposed. The proposed coding strategy can significantly reduce the transmission overhead and decoding threshold, at the expense of a slightly higher computation load.
- **Maximum Degree Decoding Strategy.** In order to recover the desired result efficiently, we give the MDD strategy by finding the disconnected encoded sub-functions with the maximum degree constantly. Compared with the classical BP decoding strategy, our MDD strategy can avoid the unnecessary decoding cost and reduce at least $\Theta\left(\ln m/r \ln m/r\right)$ additions on average, where r represents the merging parameter and m is the number of non-linear sub-functions.
- **Minimized Total Latency.** Considering the computation, transmission and decoding latency for the whole wireless network, we propose the wireless non-linear merged-r LT coded computation (WNLMrLTCC) algorithm based on the computation complexities of non-linear sub-functions, the computation and transmission capabilities of workers, the decoding capability of master, and the channel condition. Using the WNLMrLTCC algorithm, the optimal merging parameter and sub-block size can be obtained, and the minimized total latency is achieved.

Organization: The rest of this paper is organized as follows. In Section II, the wireless distributed networks are described. Also, the classification of non-linear computational tasks and the corresponding challenges are discussed. In Section III, the proposed merged-r LT coding strategy and MDD strategy are presented for non-linear tasks. Then, the performance of the new scheme is analyzed in Section IV, in terms of the computation, transmission and decoding latency. Moreover, the total latency is minimized by designing the optimal merging parameter and sub-block size. Simulation results are shown in Section V and conclusion is finally presented in Section VI.

Notation: The set $\{1, 2, \ldots, n\}$ is denoted as $[n]$ for $n \in \mathbb{N}$. As for non-negative sequences $\hat{g}\left(n\right)$ and $\hat{h}\left(n\right)$, we denote $\hat{h}\left(n\right) = \mathcal{O}(\hat{g}\left(n\right))$ if there exist constants $v > 0$ and $n_0 \in \mathbb{N}$ such that $\hat{h}\left(n\right) \leq v \cdot \hat{g}\left(n\right)$ for $\forall n > n_0$; $\hat{h}\left(n\right) = \Theta(\hat{g}\left(n\right))$ if $\hat{h}\left(n\right) = \mathcal{O}(\hat{g}\left(n\right))$ and $\hat{g}\left(n\right) = \mathcal{O}(\hat{h}\left(n\right))$; and $\hat{h}\left(n\right) = o(\hat{g}\left(n\right))$ if $\lim_{n\to\infty} \hat{h}\left(n\right)/\hat{g}\left(n\right) = 0$. The indicator function is denoted as $\mathbb{1}_{\{\cdot\}}$. The computed results $f_{\{\rho\}}\left(\mathbf{x}\right)$, $\tilde{f}_{\tilde{\nu}}\left(\mathbf{x}\right)$, $\tilde{f}_{d_{\max}}\left(\mathbf{x}\right)$,

TABLE I
MAIN NOTATIONS.

Notations	Semantics
n	Number of workers
m	Scale of computational tasks
δ_ν	Computation complexity of the ν^{th} non-linear sub-function
$\bar{\delta}$	Average computation complexity of non-linear sub-functions
ε_i	Packet erasure probability for worker i
r	Merging parameter
$d_{r,\tilde{\nu}}$	Degree of the $\tilde{\nu}^{\text{th}}$ encoded sub-function
$\bar{d}_r$	Average degree of encoded sub-functions
l	Size of computational sub-task
b_i	Sub-block size of worker i
$\{f_\nu(\cdot)\}_{\nu=1}^m$	Non-linear sub-functions (internal sub-functions)
$\{f_{\{\rho\}}(\cdot)\}_{\rho=1}^{m/r}$	Merged sub-functions
$\{\tilde{f}_{\tilde{\nu}}(\cdot)\}_{\tilde{\nu}=1}^{\alpha m/r}$	Encoded sub-functions
$\{\tilde{f}_{\tilde{\nu}_i}^{(i)}(\cdot)\}_{\tilde{\nu}_i=1}^{l}$	Computational sub-task of worker i
$\{\tilde{f}_{\tilde{\nu}_i,j}^{(i,j)}(\cdot)\}_{\tilde{\nu}_i,j=1}^{b_i}$	The j^{th} sub-block of worker i
μ_i^{cmp}	Computation straggling parameter of worker i
a_i^{cmp}	Computation shift parameter of worker i
μ_i^{trn}	Transmission parameter of worker i
μ_{dec}	Decoding straggling parameter of master
a_{dec}	Decoding shift parameter of master

$\tilde{f}'_{d_{\max}}(\mathbf{x})$ and $f_{\text{bs}}(\mathbf{x})$ are denoted as $f_{\{\rho\}}$, $\tilde{f}_{\tilde{\nu}}$, $\tilde{f}_{d_{\max}}$, $\tilde{f}'_{d_{\max}}$ and f_{bs} respectively for convenience. For ease of reading, we summarize the main notations used throughout this paper in Table I.

II. System Model and Problem Formulation

We consider a classical wireless master-worker network to perform distributed computing [3], as shown in Fig. 1. The whole network consists of a single master and n workers that own different computation and transmission capabilities. The goal is to complete a non-linear computational task wirelessly at the master with the help of these workers. This large-scale computational task can be decomposed into several small-scale ones [36], denoted as

$$F(\mathbf{x}) = \hat{f}(f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})), \tag{1}$$

where $\mathbf{x}$ is the input data that is broadcast to each worker by the master, $\hat{f}(\cdot)$ is the external function, $\{f_\nu(\cdot)\}_{\nu=1}^m$ are the internal sub-functions, and $F(\mathbf{x})$ is the desired output result. One sees from Eq. (1) that the system model is co-determined by the external function and internal sub-functions, which are pre-stored in this network.

In practice, these m internal sub-functions may be heterogeneous. This means the complexities of calculating different sub-functions may also be different. We use a normalized vector $\boldsymbol{\delta} = [\delta_1, \delta_2, \ldots, \delta_m]^{\text{T}}$ relative to the corresponding linear tasks to evaluate the complexities of sub-functions, i.e. $\delta_\nu = \mathbb{E}[T_{f_\nu}/T_{\text{L}}], \nu \in [m]$, where T_{f_ν} is the time to calculate $\varphi f_\nu(\mathbf{x})$, T_{L} is the time to calculate the linear task $\varphi\mathbf{x}$, and φ is a constant. For example, the complexities of calculating $\varphi f_1(\mathbf{x}) = \varphi\mathbf{x}^{\text{T}}\mathbf{x}\mathbf{x}$ and $\varphi f_2(\mathbf{x}) = \varphi e^{\mathbf{x}}$ are different. We assume the expected time spent on calculating $\varphi f_1(\mathbf{x})$ and $\varphi f_2(\mathbf{x})$ is respectively 4 times and 25 times more than that of $\varphi\mathbf{x}$. Then, the normalized computation complexities for these two non-linear sub-functions can be given as $\boldsymbol{\delta} = [4, 25]^{\text{T}}$.

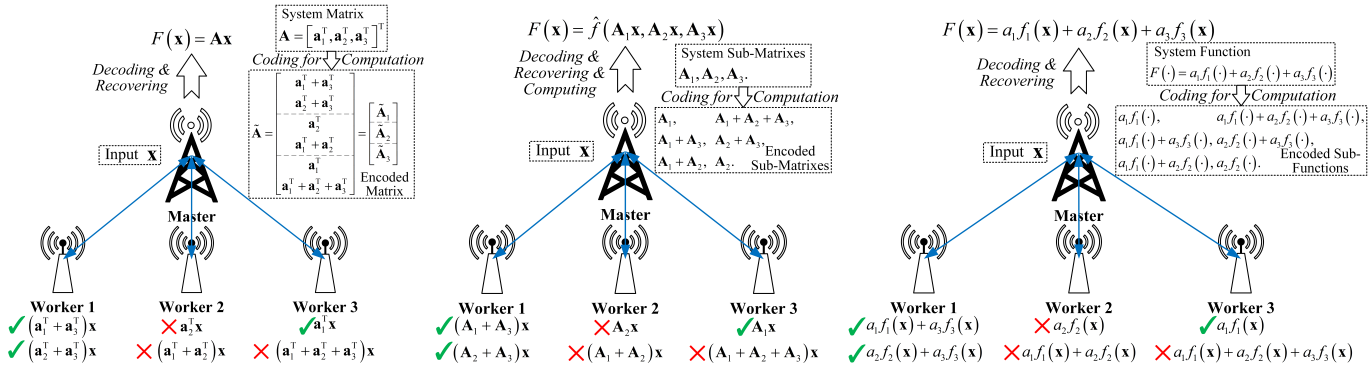
As for the wireless transmission in this network, time-division is assumed so that only one worker can transmit its packet to the master at each time[1]. Also, the worker that finishes the computation early can transmit this result early and the next one has to wait until the previous worker finishes its transmission. Due to channel fading, outage of workers and so on, transmission failure may occur. We model these failures as an erasure channel with a fixed packet erasure probability ε_i for worker i. To guarantee successful decoding, the corresponding packet will be re-transmitted in case of a transmission failure. Because of the limited bandwidth of wireless channel, a uniform maximum number of packets that can be transmitted successfully from each worker is pre-allocated to avoid too much overhead, which is denoted as a constant k_1. Furthermore, the maximum number of transmission for each packet is limited to k_2, where k_2 is also a constant. In other words, each worker can send up to k_1 packets to master and each packet can be re-transmitted at most k_2 times to prevent excessive occupation of channel resources.

Due to the composition of $\hat{f}(\cdot)$ and $\{f_\nu(\cdot)\}_{\nu=1}^m$, it is hard to analyze the properties of $F(\cdot)$ directly. In the following, we divide the non-linear computational tasks into three categories based on the features of external and internal functions[2].

- *Category 1: Both external and internal functions are linear.* In this case, the computational task degenerates to a linear task and a representative task is matrix multiplication $F(\mathbf{x}) = \mathbf{A}\mathbf{x}$ [3], [4], where $\mathbf{A}$ is the system matrix. Utilizing the classical coded matrix multiplication algorithm, the task can be solved efficiently, as shown in Fig. 1a.
- *Category 2: The external function is non-linear, while the internal functions are linear.* These tasks include the activation functions in deep neural networks (DNN) [37], the non-affine non-linear systems in control theory [38], and the non-linear feature transforms in machine learning [39] with multivariate non-linear functions $F(\mathbf{x}) = \hat{f}(\mathbf{A}_1\mathbf{x}, \ldots, \mathbf{A}_m\mathbf{x})$ where the sub-matrixes $\{\mathbf{A}_\nu\}_{\nu=1}^m$ represent the system model. In order to obtain the computed result efficiently, the internal linear functions are completed by workers with coded distributed computing and the external non-linear function is calculated by the master directly, as shown in Fig. 1b.
- *Category 3: The external function is linear, while the internal functions are non-linear.* These tasks can be seen as the linearly separable computation $F(\mathbf{x}) = \mathbf{a}^{\text{T}}\boldsymbol{f}(\mathbf{x})$ [32], where $\mathbf{a} = [a_1, \ldots, a_m]^{\text{T}}$ is the coefficient vector and $\boldsymbol{f}(\cdot) = [f_1(\cdot), \ldots, f_m(\cdot)]^{\text{T}}$ consists of non-linear sub-functions. For example, the non-linear sub-function $f_i(\cdot)$ can be regarded as the i^{th} convolution filtering operation in convolutional neural networks (CNN) [1], the i^{th} partial gradient in batch gradient descent algorithm [29], the i^{th} decision tree in random forests [40], the kernel function between the i^{th} training point and the test point in kernel methods [41] and so on. To speed up the calculation, coded computation is applied by treating a

[1]The proposed merged-r LT coded computation scheme (Alg. 1) and Alg. 2 can also be readily expanded to the case where workers are able to transmit data simultaneously, since the transmission latency analyzed in this work can be regarded as the upper bound in another case.

[2]When the external and internal functions are both non-linear, coded computation cannot be utilized directly since the exact composition relationship between external and internal functions is unclear. One way to solve these tasks is to approximate them as tasks in Category 3 using Taylor expansion.



(a) Category 1: matrix multiplication. (b) Category 2: multivariate non-linear function. (c) Category 3: linearly separable computation.

Fig. 1. Distributed coded computation for non-linear tasks in wireless networks with three cases. A total of one master and three workers complete together the calculation of the non-linear function $F(\mathbf{x})$. Worker 2 is a straggler, which slows down the whole network. With the help of coded computation, the master can recover $F(\mathbf{x})$ from the computed results of worker 1 and worker 3.

TABLE II
CLASSIFICATION OF NON-LINEAR COMPUTATIONAL TASKS.

Category	Specific Example	Applications
1	$\mathbf{Ax}$ (Fig. 1a)	Matrix-Vector Multiplication [3]; Matrix-Matrix Multiplication [4].
2	$\hat{f}(\mathbf{A}_1\mathbf{x},\ldots,\mathbf{A}_m\mathbf{x})$ (Fig. 1b)	Activation Functions [37]; Non-affine Non-Linear Systems [38]; Non-linear Feature Transforms [39].
3	$\mathbf{a}^{\mathrm{T}}\boldsymbol{f}(\mathbf{x})$ (Fig. 1c)	Convolution Filtering Operations [1]; Batch Gradient Descent [29]; Random Forests [40]; Kernel Methods [41].

non-linear sub-function as a single source symbol, as shown in Fig. 1c.

The above classification is summarized in Table II. There are many schemes to complete the tasks in Categories 1 and 2 with coded computation. Due to the space limitation, the reader is referred to [3], [4], [5], [7] and [9] for details. For Category 3, the sub-tasks allocated to workers are non-linear, heterogeneous and non-additive, which are quite different from the linear tasks in Categories 1 and 2. The following challenges emerge when dealing with these non-linear tasks.

1) *The computation and transmission costs are high.* Compared with the linear tasks, the magnitude of encoded sub-functions is much larger because the non-additivity of non-linear sub-functions precludes the pre-computation for encoding process. This leads to the high computation and transmission costs. For example, the costs of computing and transmitting $a_1\mathbf{x}^{\mathrm{T}}\mathbf{x}\mathbf{x}+a_2e^{\mathbf{x}}$ are significantly higher than those of $\left(\mathbf{a}_1^{\mathrm{T}}+\mathbf{a}_2^{\mathrm{T}}\right)\mathbf{x}$. Also, the encoded sub-functions covering more non-linear sub-functions will result in a higher computation latency. In other words, computing the encoded sub-function $f_1(\mathbf{x})+f_2(\mathbf{x})+f_3(\mathbf{x})$ is more time-consuming than computing the encoded sub-function $f_1(\mathbf{x})+f_2(\mathbf{x})$. This also implies that dense codes like MDS code are infeasible for non-linear tasks. Thus, it is critical to reduce these costs by designing an efficient coding strategy for non-linear tasks.

2) *The existing decoding strategy is inefficient.* For linear tasks, the existing BP strategy is applied to decode each inner product precisely. For the non-linear tasks in Category 3, the goal is to recover the sum, so the existing decoding strategy is inefficient and brings huge unnecessary decoding costs, which is intolerable for latency-sensitive services. A new decoding strategy to avoid these unnecessary costs is needed.

3) *The non-linear sub-functions are heterogeneous.* Non-linear tasks consist of various sub-functions. The complexities of computing different non-linear sub-functions may also be different. For example, the computation costs of $a_1\mathbf{x}^{\mathrm{T}}\mathbf{x}\mathbf{x}$ and $a_2e^{\mathbf{x}}$ are totally different. Hence, it is essential to provide an optimal allocation strategy based on the disparate computation complexities of non-linear sub-functions.

III. PROPOSED MERGED-r LT CODED COMPUTATION

In this section, we will propose a new merged-r LT coded computation scheme to solve the non-linear computational tasks efficiently. The typical large-scale linearly separable computation with non-linear sub-functions $F(\mathbf{x})=\mathbf{a}^{\mathrm{T}}\boldsymbol{f}(\mathbf{x})$ (Category 3) will be considered. To be more specific, we will first present the merged-r LT coding strategy to reduce the high computation and transmission costs. Then, the MDD strategy will be given to improve the decoding efficiency.

A. Merged-r LT Coding Strategy

To speed up the non-linear computational tasks and avoid the stragglers' effect in distributed wireless networks, the merged-r LT coding strategy is proposed based on the rateless and sparse LT code. The specific process can be described as follows.

1) Master Performs Merge Operation: For the non-linear sub-functions, the merge operation is applied before the encoding process. Specifically, master treats r different non-linear sub-functions as a source symbol, which can be given as

$$\begin{aligned} F(\cdot) &= a_1f_1(\cdot)+a_2f_2(\cdot)+\ldots+a_mf_m(\cdot) \\ &= \sum_{\nu=1}^{r}a_\nu f_\nu(\cdot)+\ldots+\sum_{\nu=m-r+1}^{m}a_\nu f_\nu(\cdot)=\sum_{\rho=1}^{m/r}f_{\{\rho\}}(\cdot), \end{aligned} \quad (2)$$

where $f_{\{\rho\}}(\cdot)=\sum_{\nu=\rho r-r+1}^{\rho r}a_\nu f_\nu(\cdot), \rho\in[m/r]$ is the merged sub-function, and r is the merging parameter representing the number of non-linear sub-functions in $f_{\{\rho\}}(\cdot)$.

2) Master Performs Encoding: The merged sub-functions are encoded by the master with LT code to obtain the encoded sub-functions $\{\tilde{f}_{\tilde{\nu}}(\cdot)\}_{\tilde{\nu}=1}^{\alpha m/r}$ as

$$\tilde{f}_{\tilde{\nu}}(\cdot) = \sum_{\tilde{\rho}\in\mathcal{S}_{\tilde{\nu}}^{d_{r,\tilde{\nu}}}} f_{\{\tilde{\rho}\}}(\cdot), \tilde{\nu} \in \left[\frac{\alpha m}{r}\right], \tag{3}$$

where $\alpha > 1$ is the encoding parameter, and the set $\mathcal{S}_{\tilde{\nu}}^{d_{r,\tilde{\nu}}}$ represents the corresponding $d_{r,\tilde{\nu}}$ merged sub-functions with uniform random selection in $\tilde{f}_{\tilde{\nu}}(\cdot)$. Moreover, $d_{r,\tilde{\nu}}$ is the degree in merged-r LT coding strategy representing the number of merged sub-functions in an encoded sub-function $\tilde{f}_{\tilde{\nu}}(\cdot)$ and reflecting the sparsity of code. The degree is determined by the robust soliton degree distribution so that the average degree of encoded sub-functions $\bar{d}_r$ is linear with $\ln m/r$ [10], [42] as

$$\bar{d}_r = w_1 \ln \frac{m}{r} + w_2, \tag{4}$$

where w_1 and w_2 are constants determined by the degree distribution.

3) Master Performs Sub-Block Division: After the encoding process, in order to carry out computations in a distributed manner by workers, through dividing $\{\tilde{f}_{\tilde{\nu}}(\cdot)\}_{\tilde{\nu}=1}^{\alpha m/r}$ equally and uniformly, the computational sub-task for each worker $\{\tilde{f}_{\tilde{\nu}_i}^{(i)}(\cdot)\}_{\tilde{\nu}_i=1}^{l}, i \in [n]$ can be obtained by the master, where $l = \alpha m/rn$ represents the number of encoded sub-functions in each computational sub-task. Each sub-task should possess the similar total degree and computation complexity[3]. To utilize the rateless property of LT code and the partial works done by stragglers, the computational sub-task for worker $i \in [n]$ is divided again by the master into sub-blocks of the same size as $\{\tilde{f}_{\tilde{\nu}_{i,j}}^{(i,j)}(\cdot)\}_{\tilde{\nu}_{i,j}=1}^{b_i}, j \in [l/b_i]$, where b_i denotes the sub-block size, i.e., each sub-block includes b_i encoded sub-functions or $\sum_{\tilde{\nu}_{i,j}=1}^{b_i} r d_{r,\tilde{\nu}_{i,j}}$ non-linear sub-functions to be calculated. Similarly, each sub-block should ideally possess the same average degree and computation complexity. In other words, the master ensures

$$\frac{\sum_{\tilde{\nu}_{i,j}=1}^{b_i} d_{r,\tilde{\nu}_{i,j}}}{b_i} \approx \frac{r\sum_{\tilde{\nu}=1}^{\alpha m/r} d_{r,\tilde{\nu}}}{\alpha m}, \tag{5}$$

$$\frac{\sum_{\tilde{\nu}_{i,j}=1}^{b_i}\sum_{\nu=1}^{r d_{r,\tilde{\nu}_{i,j}}} \delta_\nu}{\sum_{\tilde{\nu}_{i,j}=1}^{b_i} d_{r,\tilde{\nu}_{i,j}}} \approx \frac{\sum_{\tilde{\nu}=1}^{\alpha m/r}\sum_{\nu=1}^{r d_{r,\tilde{\nu}}} \delta_\nu}{\sum_{\tilde{\nu}=1}^{\alpha m/r} d_{r,\tilde{\nu}}}, \tag{6}$$

when generating sub-blocks[3]. Then, the master sends the sub-blocks and input to the corresponding workers.

[3]As a feasible way to perform the sub-task and sub-block division, the master first classifies all the encoded sub-functions into four categories based on their deviation from the average degree and computation complexity: (i) large degree and high computation complexity; (ii) small degree and low computation complexity; (iii) large degree and low computation complexity; (iv) small degree and high computation complexity. Then, the master can combine encoded sub-functions (i) with encoded sub-functions (ii), and encoded sub-functions (iii) with encoded sub-functions (iv), respectively, to generate sub-tasks and sub-blocks with the similar average degree and computation complexity. Besides, for sub-block division, if the sub-block size is large, the master will preferentially choose the encoded sub-functions that deviate from the average; if the sub-block size is small, the master will preferentially choose the encoded sub-functions around the average.

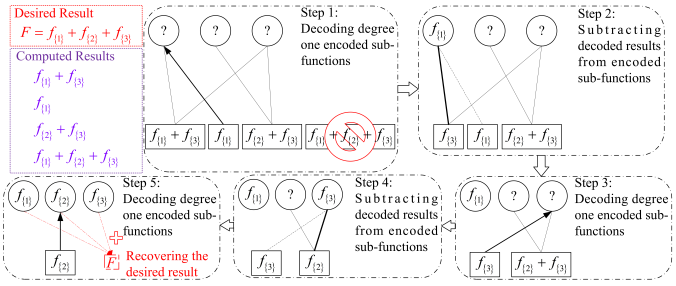


Fig. 2. The workflow of BP strategy. BP strategy is performed in an iterative fashion. In each iteration, master decodes the degree one encoded sub-functions, and subtracts the decoded results from all other encoded sub-functions connected to those decoded results. Using BP strategy, every computed result of merged sub-function is recovered exactly. $N_{\text{dec}}^{\text{BP}} = 4$.

4) Workers Perform Computing and Transmitting: Workers will begin calculating once receiving $\mathbf{x}$. When a sub-block is computed completely, workers will transmit the corresponding result to the master as soon as possible.

5) Master Performs Decoding: Based on the rateless property of LT code, the master can decode $F(\mathbf{x})$ successfully after only receiving $(1+\eta_{\text{LT}})\, m/r$ computed results of encoded sub-functions, where η_{LT} is a small decoding overhead[4] ($\eta_{\text{LT}} \to 0$ as $m \to \infty$).

According to the preceding analysis, we can observe that the proposed coding strategy can significantly reduce the high computation costs of encoded sub-functions caused by non-linearity, because the sparsity of LT code limits the magnitude of encoded sub-functions. Moreover, the merge operation can reduce not only the transmission overhead but also the decoding threshold, possibly at the expense of a slightly higher computation load. It implies that the merging parameter r has a significant impact on the computation, transmission and decoding latency for the whole network. Thus, it is essential to choose r carefully. This will be discussed in Section IV-D.

B. Maximum Degree Decoding Strategy

For the decoding process, the classical BP strategy recovers each non-linear sub-function $f_\nu(\mathbf{x}), \nu \in [m]$ exactly through iterative peeling algorithm [42], as shown in Fig. 2. This incurs a huge amount of unnecessary decoding costs in BP strategy, which may cause severe performance degradation for the whole network. Thus, based on the property of non-linear tasks in Category 3, we propose the MDD strategy to speed up the decoding process, as shown in Fig. 3. To facilitate the understanding, we first give the following definition and example.

Definition 1 (Connection). For the proposed merged-r LT code, the connection of $f_{\{\rho\}}(\mathbf{x})$, denoted as ψ_ρ, is the total number of encoded sub-functions which cover this merged sub-function. If the computed results of two different encoded sub-functions do not cover the same merged sub-function, these two encoded sub-functions are ***disconnected*** with each other. On the other hand, if they share at least one same merged

[4]In this paper, the value of η_{LT} is assumed to be fixed, since a large-scale computational task is considered.

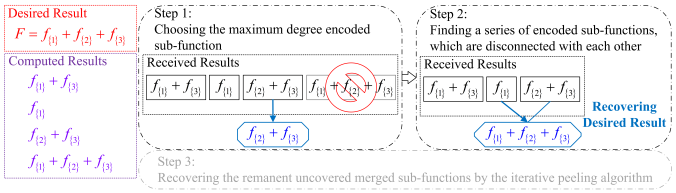


Fig. 3. The workflow of MDD strategy. In order to reduce the decoding cost of non-linear coded computation, not every computed result of merged sub-function is recovered exactly. $N_{\text{dec}}^{\text{MDD}} = 1$.

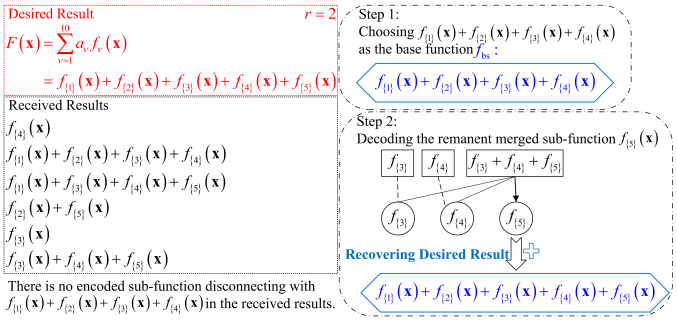


Fig. 4. A simple example of merged-r LT coding strategy and MDD strategy with the desired result $F(\mathbf{x}) = \sum_{\nu=1}^{10} f_\nu(\mathbf{x}) = f_{\{1\}} + f_{\{2\}} + f_{\{3\}} + f_{\{4\}} + f_{\{5\}}$ ($r = 2$). From six received results, $f_{\{1\}} + f_{\{2\}} + f_{\{3\}} + f_{\{4\}}$ is chosen as the base function. Due to no disconnected encoded sub-function, the remanent merged sub-function $f_{\{5\}}$ is decoded by iterative peeling algorithm. Hence $F(\mathbf{x})$ is recovered. $N_{\text{dec}}^{\text{MDD}} = 3$.

sub-function, they are ***connected*** with each other. Moreover, the more same merged sub-functions they share, the stronger ***connections*** they can achieve.

Example 1. As illustrated in Fig. 3, there are four computed results $f_{\{1\}} + f_{\{3\}}$, $f_{\{1\}}$, $f_{\{2\}} + f_{\{3\}}$, $f_{\{1\}} + f_{\{2\}} + f_{\{3\}}$. The connection of $f_{\{1\}}$ is $\psi_1 = 3$ and the connection of $f_{\{2\}}$ is $\psi_2 = 2$. $f_{\{1\}}$ is disconnected with $f_{\{2\}} + f_{\{3\}}$, while $f_{\{1\}} + f_{\{3\}}$ and $f_{\{2\}} + f_{\{3\}}$ are connected with each other.

Then, the specific process of MDD strategy can be described as follows:

Step 1*: Choose the computed result of encoded sub-function with the maximum degree $\tilde{f}_{d_{\max}}$ as a base function* f_{bs}. If $\tilde{f}_{d_{\max}}$ is unique, $f_{\text{bs}} = \tilde{f}_{d_{\max}}$. If $\tilde{f}_{d_{\max}}$ is not unique, choose the one covering a merged sub-function with a lower connection. For example, when $\deg(\tilde{f}_1) = \deg(\tilde{f}_2) = d_{\max}$, $f_{\text{bs}} = \tilde{f}_1$ if $\min\{\psi_\rho | \rho \in \mathcal{S}_1^{d_{\max}}\} < \min\{\psi_{\rho'} | \rho' \in \mathcal{S}_2^{d_{\max}}\}$;

Step 2*: Find the computed result of encoded sub-function with the maximum degree $\tilde{f}'_{d_{\max}}$ that is disconnected with* f_{bs}. *Then, add $\tilde{f}'_{d_{\max}}$ to* f_{bs}, *i.e.* $f_{\text{bs}} \leftarrow f_{\text{bs}} + \tilde{f}'_{d_{\max}}$. Similarly, if $\tilde{f}'_{d_{\max}}$ is not unique, choose the one covering a merged sub-function with a lower connection. Repeat until there is no computed result of encoded sub-function that is disconnected with f_{bs}. Then, if f_{bs} covers all the results of merged sub-functions, the desired result $F(\mathbf{x}) = f_{\text{bs}}$ will be recovered; otherwise, continue to *Step* 3;

Step 3*: Utilize iterative peeling algorithm to recover the remanent results of merged sub-functions that are not covered by* f_{bs}. In other words, based on the low-degree received results, decrease the degree of encoded sub-functions covering the remanent merged sub-functions until all the remanent merged sub-functions are recovered. Add these remanent merged sub-functions to f_{bs}, the desired result $F(\mathbf{x})$ is recovered.

The following example of MDD strategy is given to facilitate the understanding.

Example 2. As illustrated in Fig. 4, the desired non-linear computational task is $F(\mathbf{x}) = \sum_{\nu=1}^{10} a_\nu f_\nu(\mathbf{x})$. Using the merged-$r$ LT coding strategy in Section III-A, the corresponding encoded sub-functions are generated, where the merging parameter is chosen as $r = 2$. After computation and transmission, the master receives six computed results of encoded sub-functions. They are $f_{\{4\}}$, $f_{\{1\}} + f_{\{2\}} + f_{\{3\}} + f_{\{4\}}$, $f_{\{1\}} + f_{\{3\}} + f_{\{4\}} + f_{\{5\}}$, $f_{\{2\}} + f_{\{5\}}$, $f_{\{3\}}$, and $f_{\{3\}} + f_{\{4\}} + f_{\{5\}}$. Then, the decoding process begins, which can be described as follows:

1) *Master chooses $f_{\{1\}} + f_{\{2\}} + f_{\{3\}} + f_{\{4\}}$ with degree $d_{r,\tilde{\nu}} = 4$ as the base function* f_{bs}. Although the degree of $f_{\{1\}} + f_{\{3\}} + f_{\{4\}} + f_{\{5\}}$ is also $d_{r,\tilde{\nu}} = 4$, $f_{\{1\}} + f_{\{2\}} + f_{\{3\}} + f_{\{4\}}$ covers two merged sub-functions whose connections are both two, i.e. $f_{\{1\}}$ and $f_{\{2\}}$. Since $f_{\{1\}} + f_{\{3\}} + f_{\{4\}} + f_{\{5\}}$ only covers $f_{\{1\}}$, it is not chosen as f_{bs};

2) *Master then tries to find the computed results of encoded sub-functions that are disconnected with* f_{bs}. In this example, there is no result disconnected with $f_{\{1\}} + f_{\{2\}} + f_{\{3\}} + f_{\{4\}}$;

3) *Master uses iterative peeling algorithm to recover the remanent $f_{\{5\}}$ that is not covered by* f_{bs}. To be more specific, master decreases the degree of $f_{\{3\}} + f_{\{4\}} + f_{\{5\}}$ based on the received results $f_{\{3\}}$ and $f_{\{4\}}$ with degree $d_{r,\tilde{\nu}} = 1$. Hence $f_{\{5\}}$ is obtained;

4) *Finally, master adds $f_{\{5\}}$ to* f_{bs}, *and the desired result $f_{\{1\}} + f_{\{2\}} + f_{\{3\}} + f_{\{4\}} + f_{\{5\}}$ is recovered.*

We denote $N_{\text{dec}}^{\text{BP}}$ and $N_{\text{dec}}^{\text{MDD}}$ as the total number of decoding operations to recover the desired result $F(\mathbf{x})$ for BP strategy and MDD strategy, respectively. For example, $N_{\text{dec}}^{\text{BP}} = 4$ in Fig. 2, while $N_{\text{dec}}^{\text{MDD}} = 1$ in Fig. 3. Observe that $N_{\text{dec}}^{\text{BP}}$ and $N_{\text{dec}}^{\text{MDD}}$ are only determined by the received results. To show the advantages of the proposed decoding strategy intuitively, the complexities of BP strategy and MDD strategy are compared in the following.

Lemma 1. For a non-linear computational task with the merged-r LT coding strategy and BP strategy, $\mathbb{E}\left[N_{\text{dec}}^{\text{BP}}\right]$ can be given as

$$\mathbb{E}\left[N_{\text{dec}}^{\text{BP}}\right] = \frac{m}{r}\left(w_1 \ln \frac{m}{r} + w_2\right) - 1. \tag{7}$$

Proof. Through iterative peeling algorithm, BP strategy decreases the degrees of encoded sub-functions in each iteration. For every degree reduction of each encoded sub-function, master needs one addition. When the degree of encoded sub-function is reduced to one, the corresponding merged sub-function is recovered. It means that there are $\bar{d}_r - 1$ additions to decode a merged sub-function from the expected point of view. For recovering m/r merged sub-functions, master needs a total of $m\left(\bar{d}_r - 1\right)/r$ additions. At last, master sums these merged sub-functions up to obtain the desired result, which needs $m/r - 1$ additions. Thus, there is an average of $m\bar{d}_r/r - 1$ additions in total. ■

Different from BP strategy, the decoding operations in the proposed MDD strategy are closely related to the received results. To visually illustrate the superiority of MDD strategy, we will conduct an analysis in the worst case as follows.

Lemma 2. For a non-linear computational task with the merged-r LT coding strategy and MDD strategy, $\mathbb{E}\left[N_{\rm dec}^{\rm MDD}\right]$ is bounded by

$$\mathbb{E}\left[N_{\rm dec}^{\rm MDD}\right] \leq \left(\frac{m}{r} - w_1 \ln \frac{m}{r} - w_2\right)\left(w_1 \ln \frac{m}{r} + w_2\right). \quad (8)$$

Proof. In the worst case, there is no encoded sub-function disconnected with the base function, like Example 2. At this point, every merged sub-function is recovered through iterative peeling algorithm except the one covered by base function. In other words, there are a total of $m/r - \bar{d}_{\max}$ merged sub-functions that need to be recovered through iterative peeling algorithm from the expected point of view, where $\bar{d}_{\max}$ is the expectation of maximum degree in $(1+\eta_{\rm LT})\, m/r$ encoded sub-functions. Similarly, in order to decode these merged sub-functions successfully, master needs $\left(m/r - \bar{d}_{\max}\right)\left(\bar{d}_r - 1\right)$ additions. Then, the sum of these $m/r - \bar{d}_{\max}$ merged sub-functions and base function is obtained to recover the desired result. Thus, for the worst case, there is an average of $\left(m/r - \bar{d}_{\max}\right)\bar{d}_r$ additions during the whole decoding process, which yields Eq. (8). ∎

Then, the following proposition is given to show the performance gain of MDD strategy.

Proposition 1 (*Performance Gap between MDD Strategy and BP Strategy*). For a non-linear task with the merged-r LT coding strategy, MDD strategy can decrease at least an average of $\bar{d}_r^2 - 1$ additions compared with BP strategy, i.e.,

$$\mathbb{E}\left[N_{\rm dec}^{\rm BP}\right] - \mathbb{E}\left[N_{\rm dec}^{\rm MDD}\right] \geq \Theta\left(\ln \frac{m}{r} \ln \frac{m}{r}\right). \quad (9)$$

Proof. Based on Lemma 1 and Lemma 2, Eq. (9) can be obtained by subtracting Eq. (8) from Eq. (7). ∎

Corollary 1 (*Worst Point of View*). From the worst point of view, the performance gap between MDD strategy and BP strategy can be given as

$$N_{\rm dec,b}^{\rm BP} - N_{\rm dec,b}^{\rm MDD} = \Theta\left(\frac{m}{r} \ln \frac{m}{r}\right), \quad (10)$$

where $N_{\rm dec,b}^{\rm BP}$ and $N_{\rm dec,b}^{\rm MDD}$ are the total number of decoding operations to recover the desired result $F(\mathbf{x})$ for BP strategy and MDD strategy, respectively, from the worst point of view.

Proof. See Appendix A. ∎

In order to thoroughly illustrate the superiority of MDD strategy, the performance comparisons of different decoding strategies are presented in Fig. 5[5] (decoding latency will be discussed in Sec. IV-C). From both calculated amount's and latency's points of view, the decoding performance gap between MDD strategy and the other decoding strategies will widen as the scale of computational tasks m increases. Moreover, MDD strategy even in the worst case performs better than BP strategy, which confirms the above theoretical analysis.

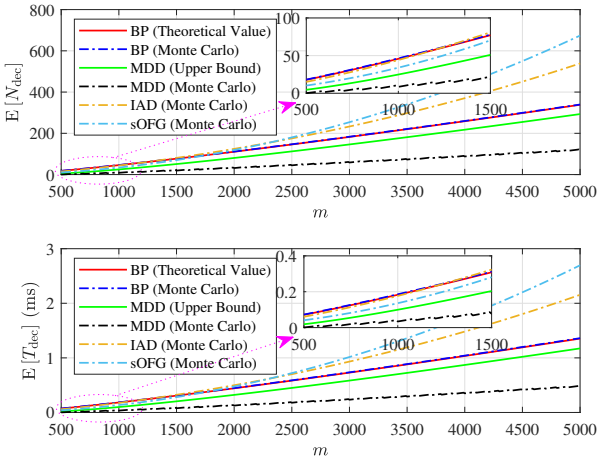


Fig. 5. Performance comparisons between IAD strategy [43], sOFG strategy [44], BP strategy, and MDD strategy with m, where $r = 100$, $\mu_{\rm dec} = 500$, and $a_{\rm dec} = 0.002$.

We summarize our proposed merged-r LT coded computation scheme including merged-r LT coding strategy and MDD strategy as Alg. 1.

IV. Latency Analysis and Optimization

For the non-linear computational tasks in Category 3, different coding and decoding strategies lead to significantly different latency performance in wireless networks. In this section, we will first analyze the computation, transmission and decoding latency of the proposed merged-r LT coded computation. Also, the factors that influence the latency will be discussed. At last, in order to minimize the total latency, the merging parameter and sub-block size will be optimized.

[5]The inactivation decoding (IAD) strategy was thoroughly analyzed in [43], and the soft on-the-fly Gaussian elimination (sOFG) decoding strategy was proposed in [44]. In IAD strategy, the iterative decoding algorithm is applied first, resulting in some computed results that cannot be recovered. At the end of the decoding process, these unrecovered results are solved using Gaussian elimination algorithm. IAD strategy can reduce $\eta_{\rm LT}$ at the expense of a higher decoding cost. Moreover, sOFG strategy is also designed based on Gaussian elimination algorithm, which can further decrease $\eta_{\rm LT}$. In sOFG strategy, although the master can perform some decoding pre-processing while receiving computed results, it still leads to an increased decoding cost. In general, IAD strategy and sOFG strategy can achieve good performance when m is small, while BP strategy is more effective for large-scale computational tasks.

[6]The optimal merging parameter r^* and sub-block size $\{b_i^*\}_{i=1}^n$ are obtained by Alg. 2.

Algorithm 1 Merged-r LT Coded Computation Scheme[6]

Require: The coefficient vector $\mathbf{a} = [a_1, \ldots, a_m]^{\mathrm{T}}$, non-linear sub-functions $\boldsymbol{f}(\cdot) = [f_1(\cdot), \ldots, f_m(\cdot)]^{\mathrm{T}}$, and input data $\mathbf{x}$.

Ensure: The desired output result $F(\mathbf{x})$.

procedure Merged-r LT Coded Computation Scheme
  Master performs *Merged-r LT Coding Strategy*:
    Obtain the merged sub-functions $f_{\{\rho\}}(\cdot)$ in Eq. (2) for $\rho \in [m/r]$;
    Obtain the encoded sub-functions $\tilde{f}_{\tilde{\nu}}(\cdot)$ in Eq. (3) for $\tilde{\nu} \in [\alpha m/r]$;
    Obtain the sub-blocks $\{\tilde{f}^{(i,j)}_{\tilde{\nu}_{i,j}}(\cdot)\}^{b_i}_{\tilde{\nu}_{i,j}=1}$ based on the division strategies (5) and (6) for $j \in [l/b_i]$ and $i \in [n]$;
    Send sub-blocks to workers;
  Master broadcasts $\mathbf{x}$ to workers;
  Workers computes $\{\tilde{f}^{(i,j)}_{\tilde{\nu}_{i,j}}(\mathbf{x})\}^{b_i}_{\tilde{\nu}_{i,j}=1}, j \in [l/b_i], i \in [n]$, and transmits these results to master;
  if master receives more than $(1+\eta_{\mathrm{LT}})\, m/r$ computed results of encoded sub-functions **then**
    Master performs *MDD Strategy*:
      Obtain f_{bs} according to *Step* 1;
      repeat
        Obtain $\tilde{f}'_{d_{\max}}$ according to *Step* 2;
        Set $f_{\mathrm{bs}} \leftarrow f_{\mathrm{bs}} + \tilde{f}'_{d_{\max}}$;
      until no disconnected encoded sub-function;
      Recover the remanent uncovered results according to *Step* 3;
      return $F(\mathbf{x})$ by adding these remanent uncovered results to f_{bs}.
  end if
end procedure

A. Computation Latency

As for the non-linear tasks with merged-r LT coded computation, the following definition is given to evaluate the computation latency for the whole network.

Definition 2 (**Computation Latency**)**.** For the non-linear computational tasks using the proposed merged-r LT coding strategy and MDD strategy, the computation latency, denoted as T_{cmp}, is the time spent on calculating $(1+\eta)\, m/r$ encoded sub-functions for the whole network, where η is the extra overhead caused by LT decoding and transmission failure. It is a random variable, given as:

$$T_{\mathrm{cmp}} = \max_{i \in [n]} T_i^{\mathrm{cmp}}, \tag{11}$$

where T_i^{cmp} is also a random variable representing the total computation time for worker i, and all the random variables $\{T_i^{\mathrm{cmp}}\}_{i=1}^n$ are assumed to be mutually independent.

Based on the sub-block division strategies (5) and (6), we assume that the computation load for each sub-block is directly proportional to its size to facilitate the analysis, i.e.,

$$\sum_{\tilde{\nu}_{i,j}=1}^{b_i} \sum_{\nu=1}^{r d_{r,\tilde{\nu}_{i,j}}} \delta_\nu = b_i r \bar{d}_r \bar{\delta}, j \in \left[\frac{l}{b_i}\right], i \in [n], \tag{12}$$

since the average degree and computation complexity of sub-blocks for all the n workers are the same, where $\bar{\delta}$ is the average computation complexity of non-linear sub-functions. This implies the cost of computing a single sub-block is equal to that of computing $b_i r \bar{d}_r$ non-linear sub-functions with the same computation complexity $\bar{\delta}$, for worker i. Then, the time of computing j sub-blocks for worker i, i.e. jb_i encoded sub-functions, is denoted as a random variable $T_{i,j}^{\mathrm{cmp}}$. The cumulative distribution function (CDF) of $T_{i,j}^{\mathrm{cmp}}$ can be described as a shifted exponential distribution [3]:

$$\Pr\left[T_{i,j}^{\mathrm{cmp}} \le t\right] = 1 - e^{-\frac{\mu_i^{\mathrm{cmp}}}{j b_i \bar{\delta} r \bar{d}_r}\left(t - j b_i \bar{\delta} r \bar{d}_r a_i^{\mathrm{cmp}}\right)}, \tag{13}$$

for $t \ge j b_i \bar{\delta} r \bar{d}_r a_i^{\mathrm{cmp}}$ and $j \le k_1$, where μ_i^{cmp} and a_i^{cmp} are the straggling and shift parameters, respectively, representing the computation capability of worker i. Assume that c_i is the number of sub-blocks computed by worker i completely before completing a total of $(1+\eta)\, m/r$ encoded sub-functions in the networks. Then, according to Eq. (13), we can observe that

$$T_i^{\mathrm{cmp}} = c_i b_i \bar{\delta} r \bar{d}_r \left(\hat{T}_i^{\mathrm{cmp}} + a_i^{\mathrm{cmp}}\right), \tag{14}$$

where the random variable $\hat{T}_i^{\mathrm{cmp}}$ represents the initial setup time at worker i before actually beginning the calculation, following an exponential distribution with rate parameter μ_i^{cmp}.

Due to the sub-block division and limited transmission rounds, the number of sub-blocks calculated by worker i with a given computation time t, denoted as $x_i(t)$, is given by

$$x_i(t) = \begin{cases} 0, & \text{if } t < T_{i,1}^{\mathrm{cmp}}, \\ j, & \text{if } T_{i,j}^{\mathrm{cmp}} \le t < T_{i,j+1}^{\mathrm{cmp}},\ j \in [k_1 - 1], \\ k_1, & \text{if } t \ge T_{i,k_1}^{\mathrm{cmp}}. \end{cases} \tag{15}$$

Thus, through the definition of the mean, Eq. (13) and Eq. (15), the expected number of sub-blocks calculated by worker i can be derived as

$$\begin{aligned}
&\mathbb{E}\left[x_i(t)\right] = \sum_{j=0}^{k_1} j \times \Pr\left[x_i(t) = j\right] \\
&= \sum_{j=1}^{k_1-1} j \times \Pr\left[T_{i,j}^{\mathrm{cmp}} \le t < T_{i,j+1}^{\mathrm{cmp}}\right] + k_1 \times \Pr\left[t \ge T_{i,k_1}^{\mathrm{cmp}}\right] \\
&= \sum_{j=1}^{k_1-1} j \times \left(\Pr\left[T_{i,j}^{\mathrm{cmp}} \le t\right] - \Pr\left[T_{i,j+1}^{\mathrm{cmp}} \le t\right]\right) \\
&\quad + k_1 \times \Pr\left[T_{i,k_1}^{\mathrm{cmp}} \le t\right] \\
&= \sum_{j=1}^{k_1} \Pr\left[T_{i,j}^{\mathrm{cmp}} \le t\right] \\
&= \sum_{j=1}^{k_1} \left(1 - e^{-\frac{\mu_i^{\mathrm{cmp}}}{j b_i \bar{\delta} r \bar{d}_r}\left(t - j b_i \bar{\delta} r \bar{d}_r a_i^{\mathrm{cmp}}\right)}\right).
\end{aligned} \tag{16}$$

Based on $c_i = x_i\left(T_{\rm cmp}\right)$, we can obtain

$$\mathbb{E}\left[c_i\right] = \sum_{j=1}^{k_1}\left(1 - e^{-\frac{\mu_i^{\rm cmp}}{jb_i\bar{\delta}r\bar{d}_r}\left(T_{\rm cmp} - jb_i\bar{\delta}r\bar{d}_r a_i^{\rm cmp}\right)}\right). \tag{17}$$

For the heterogeneous distributed networks with merged-r LT coded computation, the order statistics cannot be used to describe the computation latency directly because of sub-block division and disparate capabilities of workers, so the exact expression of $\mathbb{E}\left[T_{\rm cmp}\right]$ is hard to obtain. Hence, we discuss the bounds of computation latency in the following proposition.

Proposition 2 (*The Bounds of Computation Latency*). Using merged-r LT coding strategy and MDD strategy for non-linear tasks, the expected upper and lower bounds of computation latency for the whole network can be given respectively as

$$\mathbb{E}\left[T_{\rm cmp}\right] \le \bar{\delta}\bar{d}_r\Big(\frac{\alpha m}{n} + r\Big)\Big(\frac{1}{\mu_{\rm b}^{\rm cmp}} + a_{\rm b}^{\rm cmp}\Big), \tag{18}$$

$$\mathbb{E}\left[T_{\rm cmp}\right] \ge \frac{\left(1+\eta_{\rm LT}\right)m}{\left(1-\varepsilon_{\rm g}^{k_2}\right)n}\bar{\delta}\bar{d}_r\Big(\frac{1}{n\mu_{\rm g}^{\rm cmp}} + a_{\rm g}^{\rm cmp}\Big), \tag{19}$$

where $\left\{\mu_{\rm b}^{\rm cmp}, a_{\rm b}^{\rm cmp}\right\} = \arg\max_i\left(1/\mu_i^{\rm cmp} + a_i^{\rm cmp}\right)$, $\varepsilon_{\rm g} = \min_i\varepsilon_i$ and $\left\{\mu_{\rm g}^{\rm cmp}, a_{\rm g}^{\rm cmp}\right\} = \arg\min_i\left(1/\mu_i^{\rm cmp} + a_i^{\rm cmp}\right)$ for $i \in [n]$.

Proof. See Appendix B. ∎

B. Transmission Latency

As for the non-linear tasks with merged-r LT coded computation, the following definition is given to evaluate the transmission latency for the whole network.

Definition 3 (**Transmission Latency**)**.** For the non-linear computational tasks using the proposed merged-r LT coding strategy and MDD strategy, the transmission latency, denoted as $T_{\rm trn}$, is the time spent on transmitting all the computational tasks calculated by n workers completely for the whole network, i.e. $\left(1+\eta\right)m/r$ computed results of encoded sub-functions. It is a random variable that is highly related to computation latency, given as:

$$T_{\rm trn} = \sum_{i=1}^{n} T_i^{\rm trn}, \tag{20}$$

where $T_i^{\rm trn}$ represents the total transmission time for worker i including re-transmission.

Due to the instability and limited bandwidth of wireless channel, workers may re-transmit the computed encoded sub-function many times. It is assumed that the time spent on transmitting the result of the $j'^{\rm th}$ encoded sub-function during the $\kappa^{\rm th}$ transmission for worker i, denoted as $T_{i,j',\kappa}^{\rm trn}$, follows a mutually independent exponential distribution [6], i.e.,

$$\Pr\left[T_{i,j',\kappa}^{\rm trn} \le t\right] = 1 - e^{-\mu_i^{\rm trn}t}, \tag{21}$$

where the rate parameter $\mu_i^{\rm trn}$ represents the transmission capability of worker i. Then, the total transmission time for worker i can be obtained by

$$T_i^{\rm trn} = \sum_{j'=1}^{c_i b_i}\sum_{\kappa=1}^{K_i^{\rm trn}} T_{i,j',\kappa}^{\rm trn} = \sum_{j'=1}^{c_i b_i} T_{i,j'}^{\rm trn}, \tag{22}$$

where $T_{i,j'}^{\rm trn} = \sum_{\kappa=1}^{K_i^{\rm trn}} T_{i,j',\kappa}^{\rm trn}$ is the total time for the result of the $j'^{\rm th}$ encoded sub-function transmitted by worker i, and $K_i^{\rm trn}$ is the number of the corresponding total transmission times, which can be given as

$$\Pr\left[K_i^{\rm trn} = \kappa\right] = \begin{cases}\varepsilon_i^{\kappa-1}\left(1-\varepsilon_i\right), & \text{if } \kappa \in [k_2 - 1],\\ \varepsilon_i^{k_2-1}, & \text{if } \kappa = k_2.\end{cases} \tag{23}$$

Hence, we can describe the transmission latency for the whole network in the following proposition.

Proposition 3 (*Transmission Latency*). Using merged-r LT coding strategy and MDD strategy for non-linear tasks, the expected transmission latency for the whole network can be given as

$$\mathbb{E}\left[T_{\rm trn}\right] = \sum_{i=1}^{n}\frac{\left(1-\varepsilon_i^{k_2}\right)b_i}{\left(1-\varepsilon_i\right)\mu_i^{\rm trn}}\mathbb{E}\left[c_i\right]. \tag{24}$$

Proof. See Appendix C. ∎

C. Decoding Latency

As for the non-linear tasks with merged-r LT coded computation, the following definition is given to evaluate the decoding latency for the whole network.

Definition 4 (**Decoding Latency**)**.** For the non-linear computational tasks using the proposed merged-r LT coding strategy and MDD strategy, the decoding latency, denoted as $T_{\rm dec}$, is the time spent on decoding $\left(1+\eta_{\rm LT}\right)m/r$ computed results of encoded sub-functions and recovering the desired result $F\left(\mathbf{x}\right)$ for the master. It is a random variable, given as:

$$T_{\rm dec} = N_{\rm dec}^{\rm MDD}T_{\rm dec}', \tag{25}$$

where $T_{\rm dec}'$ is also a random variable that represents the time of a single addition for the master during the decoding process.

According to [23], $T_{\rm dec}'$ can be described as a shifted exponential distribution with the straggling and shift parameter tuple $\left\{\mu_{\rm dec}, a_{\rm dec}\right\}$, which represents the decoding capability of the master. Then, the CDF of $T_{\rm dec}$ can be given as

$$\Pr\left[T_{\rm dec} \le t\right] = 1 - e^{-\frac{\mu_{\rm dec}}{N_{\rm dec}^{\rm MDD}}\left(t - N_{\rm dec}^{\rm MDD}a_{\rm dec}\right)}. \tag{26}$$

Hence, we can describe the decoding latency for the whole network in the following proposition.

Proposition 4 (*Decoding Latency*). Using merged-r LT coding strategy and MDD strategy for non-linear tasks, the expected decoding latency for the whole network is given as

$$\mathbb{E}\left[T_{\rm dec}\right] = \Big(\frac{1}{\mu_{\rm dec}} + a_{\rm dec}\Big)\mathbb{E}\left[N_{\rm dec}^{\rm MDD}\right]. \tag{27}$$

For the worst decoding case, $\mathbb{E}\left[T_{\rm dec}\right]$ is bounded by

$$\mathbb{E}\left[T_{\rm dec}\right] \le \Big(\frac{1}{\mu_{\rm dec}} + a_{\rm dec}\Big)\Big(\frac{m}{r} - \bar{d}_r\Big)\bar{d}_r. \tag{28}$$

Proof. $\mathbb{E}\left[T_{\rm dec}\right]$ can be obtained by taking the expectation of Eq. (25). For the worst decoding case, the upper bound of $\mathbb{E}\left[T_{\rm dec}\right]$ is obtained by substituting Eq. (8) into Eq. (27). ∎

To confirm the theoretical analysis of latency, Fig. 6 shows the expected latency versus the merging parameter for merged-r LT coded computation. It is observed that Monte Carlo

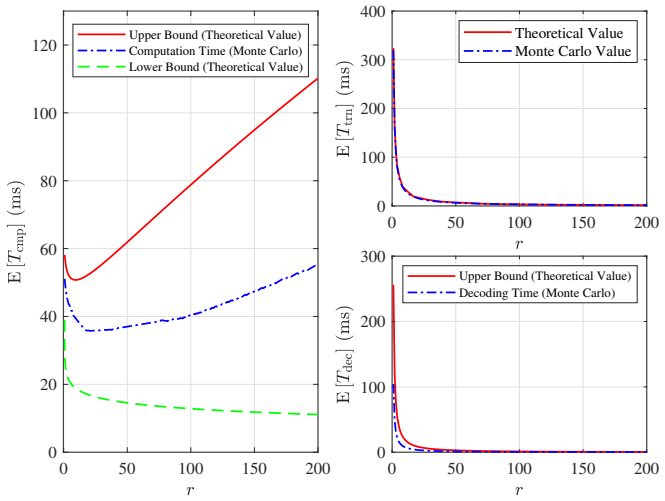


Fig. 6. Latency versus merging parameter r, where $n = 100$, $m = 5000$, $\alpha = 1.25$, $k_1 = 4$, $k_2 = 5$, $b_i = \alpha m/k_1 rn$, $\mu_i^{\mathrm{cmp}} \sim \mathcal{U}(125, 150)$, $a_i^{\mathrm{cmp}} \sim \mathcal{U}(0.009, 0.01)$, $\mu_i^{\mathrm{trn}} \sim \mathcal{U}(20, 25)$, $\varepsilon_i \sim \mathcal{U}(0, 0.5)$, $\mu_{\mathrm{dec}} = 500$, $a_{\mathrm{dec}} = 0.002$, and $\delta_\nu \sim \mathcal{U}(1, 7)$, for $i \in [n]$ and $\nu \in [m]$.

simulation results are in good agreement with the theoretical ones. As r increases, computation latency will vary, as a single encoded sub-function comprises variable non-linear sub-functions. Concurrently, transmission latency and decoding latency will decrease due to the reduction in decoding threshold and degree. It implies that there exists a trade-off between T_{cmp}, T_{trn} and T_{dec} for different r.

In order to achieve the optimal trade-off between computation, transmission and decoding latency, we consider designing the merging parameter and sub-block size, and reducing the total latency for the whole network as much as possible. Referring to [24], the total expected latency is $\mathbb{E}[T_{\mathrm{tot}}] = \mathbb{E}[T_{\mathrm{cmp}} + T_{\mathrm{trn}} + T_{\mathrm{dec}}]$. It is challenging to solve the optimization problem with merging parameter and sub-block size jointly because r and $\{b_i\}_{i=1}^n$ are coupled with each other in the exact expression of $\mathbb{E}[T_{\mathrm{tot}}]$. Similar to [8], the two-step optimization strategy is implemented. First, we design the optimal merging parameter in the worst case. It implies that the upper bound of $\mathbb{E}[T_{\mathrm{tot}}]$ is considered since the decrease in the upper bound still leads to a decrease in the total latency. Based on the obtained optimal r^*, we then formulate an optimization problem of minimizing the total latency to design the optimal sub-block size.

D. Optimal Merging Parameter

To get the optimal merging parameter, we analyze the upper bound of the total expected latency and reduce it as much as possible. As a result, r^* is given in the following proposition.

Proposition 5 (*The Optimal Merging Parameter*). Using merged-r LT coding strategy and MDD strategy, the optimal merging parameter r^* can be obtained by

$$r^* = \begin{cases} 1, & \text{if } f_{\min} = f_{\mathrm{o}}(1), \\ \tilde{r}, & \text{if } f_{\min} = f_{\mathrm{o}}(\tilde{r}), \\ m, & \text{if } f_{\min} = f_{\mathrm{o}}(m), \end{cases} \tag{29}$$

where $f_{\min} = \min\{f_{\mathrm{o}}(1), f_{\mathrm{o}}(\tilde{r}), f_{\mathrm{o}}(m)\}$, and $f_{\mathrm{o}}(r)$ is the upper bound of $\mathbb{E}[T_{\mathrm{tot}}]$, which is given as

$$f_{\mathrm{o}}(r) = s_{\mathrm{dec}}\Big(\frac{m}{r} - \bar{d}_r\Big)\bar{d}_r + \bar{\delta}\Big(\frac{\alpha m}{n} + r\Big)\bar{d}_r s_{\mathrm{b}}^{\mathrm{cmp}} \times \Big(1 + \sum_{i=1}^{n} \frac{1 - \varepsilon_i^{k_2}}{\bar{\delta} r \bar{d}_r s_i^{\mathrm{cmp}} (1 - \varepsilon_i)\, \mu_i^{\mathrm{trn}}}\Big), \tag{30}$$

where $s_{\mathrm{b}}^{\mathrm{cmp}} = 1/\mu_{\mathrm{b}}^{\mathrm{cmp}} + a_{\mathrm{b}}^{\mathrm{cmp}}$, $s_i^{\mathrm{cmp}} = 1/\mu_i^{\mathrm{cmp}} + a_i^{\mathrm{cmp}}$, $s_{\mathrm{dec}} = 1/\mu_{\mathrm{dec}} + a_{\mathrm{dec}}$, and $\tilde{r} \in [m]$ satisfies $\mathrm{d}f_{\mathrm{o}}(r)/\mathrm{d}r\,|_{r=\tilde{r}} = 0$.

Proof. See Appendix D. ∎

E. Optimal Sub-Block Size

Since the decoding latency is not affected by the sub-block size $\{b_i\}_{i=1}^n$, we can formulate the optimization problem as

$$\mathcal{P}_{\mathrm{main}}: \min_{\boldsymbol{b}} \mathbb{E}[T_{\mathrm{cmp}} + T_{\mathrm{trn}}] \quad \text{s.t. } \Pr\Big[\sum_{i=1}^{n} c_i b_i \le (1 + \eta)\frac{m}{r}\Big] = o\Big(\frac{1}{n}\Big), \tag{31}$$

where the constraint (31) ensures that the desired result can be recovered with high probability. For any given $\mu_i^{\mathrm{cmp}} > 0$, $a_i^{\mathrm{cmp}} > 0$, $\mu_i^{\mathrm{trn}} > 0$, $0 \le \varepsilon_i < 1$, $\mu_{\mathrm{dec}} > 0$, $a_{\mathrm{dec}} > 0$ and $\bar{\delta} \ge 1$, $\mathcal{P}_{\mathrm{main}}$ is always feasible because there exists at least one feasible solution, i.e., $b_i = l/k_1$ for $i \in [n]$, satisfying the constraints of $\mathcal{P}_{\mathrm{main}}$.

Due to the limited transmission rounds, as for a sub-block transmitted by worker i, we denote the probability of transmission failure after k_2 transmissions as $p_{\mathrm{f},i} = \varepsilon_i^{k_2}$ and the corresponding probability of successful transmission as $p_{\mathrm{s},i} = 1 - \varepsilon_i^{k_2}$. Then, on average, the constraint (31) can be rewritten as $\sum_{i=1}^n b_i \mathbb{E}[c_i]\, p_{\mathrm{s},i} \ge (1 + \eta_{\mathrm{LT}})\, m/r$ to ensure that $F(\mathbf{x})$ can be recovered successfully. However, due to the heavy relation between computation and transmission latency for each worker, it is still challenging to obtain the exact expression of $\mathbb{E}[T_{\mathrm{cmp}} + T_{\mathrm{trn}}]$, which makes this problem hard to solve. According to [5, Section III-A], we can reformulate the optimization problem as follows:

$$\mathcal{P}_1: \min_{t_{\mathrm{cmp}}, \boldsymbol{t}_{\mathrm{trn}}, \boldsymbol{b}} t_{\mathrm{cmp}} + \sum_{i=1}^{n} t_i^{\mathrm{trn}}$$

$$\text{s.t. } t_i^{\mathrm{trn}} \mu_i^{\mathrm{trn}} \frac{1 - \varepsilon_i}{1 - \varepsilon_i^{k_2}} \le b_i \sum_{j=1}^{k_1} \Big(1 - e^{-\frac{\mu_i^{\mathrm{cmp}}}{j b_i \bar{\delta} r \bar{d}_r}\left(t_{\mathrm{cmp}} - j b_i \bar{\delta} r \bar{d}_r a_i^{\mathrm{cmp}}\right)}\Big), \forall i \in [n] \tag{32}$$

$$\sum_{i=1}^{n} t_i^{\mathrm{trn}} \mu_i^{\mathrm{trn}} (1 - \varepsilon_i) \ge (1 + \eta_{\mathrm{LT}})\frac{m}{r}, \tag{33}$$

where the variable t_{cmp} is introduced to relax the term $\mathbb{E}[T_{\mathrm{cmp}}]$ and the set $\boldsymbol{t}_{\mathrm{trn}} = \{t_i^{\mathrm{trn}}\}_{i=1}^n$ is also the slack variable representing the transmission time of each worker. t_i^{trn} satisfies $t_i^{\mathrm{trn}} \le (1 - \varepsilon_i^{k_2}) b_i \mathbb{E}[x_i(t_{\mathrm{cmp}})] / (1 - \varepsilon_i)\mu_i^{\mathrm{trn}}$ to ensure that the number of transmitted sub-blocks is no more than the number of computed sub-blocks, which leads to the constraint (32). The constraint (33) implies that the sufficient computed results of non-linear sub-functions are aggregated to decode

Algorithm 2 Wireless Non-Linear Merged-r LT Coded Computation

Require: The parameter tuple $\{\mu_i^{\rm cmp}, a_i^{\rm cmp}, \mu_i^{\rm trn}, \varepsilon_i\}$ for each worker $i \in [n]$, $\{\mu_{\rm dec}, a_{\rm dec}\}$ for master and $\bar{\delta}$ for the non-linear computational task.

Ensure: The merging parameter r^* and sub-block size b_i^* for the worker $i \in [n]$.

1: **procedure** WNLMrLTCC
2: Obtain the optimal r^* in Eq. (29);
3: Obtain γ_i in Eq. (37) and λ_i' in Eq. (38) for the worker $i \in [n]$;
4: Obtain the optimal $t_{\rm cmp}^*$ in Eq. (34);
5: **return** r^* and $b_i^* = t_{\rm cmp}^*/\gamma_i \cdot \mathbb{1}_{\{\lambda_i'>0\}}$ for the worker $i \in [n]$.
6: **end procedure**

successfully. The solution to $\mathcal{P}_1$ is provably asymptotically optimal when n becomes very large according to [7].

For $\mathcal{P}_1$, the objective function and the constraint (33) are linear functions with respect to $t_{\rm cmp}$ and $\boldsymbol{t}_{\rm trn}$, and the constraint (32) can be rewritten as a convex exponential cone. Thus, $\mathcal{P}_1$ is a convex problem and the Karush-Kuhn-Tucker (KKT) conditions can be applied. Then, the optimal time can be obtained as follows:

$$t_{\rm cmp}^* = \frac{(1+\eta_{\rm LT})\, m}{r \sum\limits_{i=1}^{n} \frac{1-\varepsilon_i^{k_2}}{\gamma_i} h_i \cdot \mathbb{1}_{\{\lambda_i'>0\}}}, \tag{34}$$

$$t_i^{\rm trn*} = \frac{t_{\rm cmp}^* \left(1-\varepsilon_i^{k_2}\right)}{\gamma_i \left(1-\varepsilon_i\right) \mu_i^{\rm trn}} h_i \cdot \mathbb{1}_{\{\lambda_i'>0\}}, i \in [n], \tag{35}$$

where

$$h_i = k_1 - \sum_{j=1}^{k_1} e^{-\frac{\mu_i^{\rm cmp}\gamma_i}{j\bar{\delta} r \bar{d}_r} + \mu_i^{\rm cmp} a_i^{\rm cmp}}, \tag{36}$$

γ_i is the positive solution to the equation

$$-k_1 + \sum_{j=1}^{k_1} \left(1 + \frac{\mu_i^{\rm cmp}\gamma_i}{j\bar{\delta} r \bar{d}_r}\right) e^{-\frac{\mu_i^{\rm cmp}\gamma_i}{j\bar{\delta} r \bar{d}_r} + \mu_i^{\rm cmp} a_i^{\rm cmp}} = 0 \tag{37}$$

and λ_i' is the straggling factor that indicates whether the worker i is a straggler or not, which satisfies

$$\lambda_i' = \left(1-\varepsilon_i\right) \mu_i^{\rm trn} - \sum_{i'=1}^{n} g_{i'} + \sum_{i'=1}^{n} \frac{\left(1-\varepsilon_i\right) \mu_i^{\rm trn} g_{i'}}{\left(1-\varepsilon_{i'}\right) \mu_{i'}^{\rm trn}}, \tag{38}$$

where

$$g_i = \left(1-\varepsilon_i^{k_2}\right) \sum_{j=1}^{k_1} \frac{\mu_i^{\rm cmp} e^{-\frac{\mu_i^{\rm cmp}\gamma_i}{j\bar{\delta} r \bar{d}_r} + \mu_i^{\rm cmp} a_i^{\rm cmp}}}{j\bar{\delta} r \bar{d}_r}. \tag{39}$$

Moreover, the optimal sub-block size can be obtained by

$$b_i^* = \frac{t_{\rm cmp}^*}{\gamma_i} \cdot \mathbb{1}_{\{\lambda_i'>0\}}, i \in [n]. \tag{40}$$

Then, substituting the optimal merging parameter r^* into the solution to $\mathcal{P}_1$, WNLMrLTCC is provided as Alg. 2.

Using the closed-form expression of r^* and $\{b_i^*\}_{i=1}^n$, the WNLMrLTCC algorithm can be carried out in the constant time. It implies that the proposed Alg. 2 is low-complexity compared with iterative optimization algorithms like interior-point methods.

Corollary 2 (*Unlimited Transmission Rounds*). When the maximum number of sub-blocks transmitted from each worker is unlimited, i.e. k_1 is large enough, if $\mu_i^{\rm cmp} \cdot a_i^{\rm cmp}$ and l are also large, the straggling factor can be rewritten as

$$\lambda_i'' = \bar{\delta} r \bar{d}_r \left(1-\varepsilon_i\right) \mu_i^{\rm trn} - \sum_{i'=1}^{n} \frac{1-\varepsilon_{i'}^{k_2}}{a_{i'}^{\rm cmp}} + \left(1-\varepsilon_i\right) \mu_i^{\rm trn} \sum_{i'=1}^{n} \frac{1-\varepsilon_{i'}^{k_2}}{\left(1-\varepsilon_{i'}\right) \mu_{i'}^{\rm trn} a_{i'}^{\rm cmp}}, i \in [n]. \tag{41}$$

At this time, if worker i is not a straggler, i.e. $\lambda_i'' > 0$, the smaller sub-block size will lead to the better latency performance. In other words, the sub-block size obtained by WNLMrLTCC is given as $b_i^* = 1$, which degenerates to the fine-grained LT coding scheme [9]. Moreover, if each sub-block can be re-transmitted unlimitedly, i.e. $k_2 \to \infty$, η will be reduced to $\eta_{\rm LT}$.

Proof. See Appendix E. ∎

Corollary 3 (*Performance Error*). Assume that $\mu_i^{\rm cmp} \cdot a_i^{\rm cmp}$ is large. When there is an error $\Delta\bar{\delta}$ between the average computation complexity obtained by the master $\bar{\delta}$ and the actual value $\bar{\delta}'$, i.e. $\Delta\bar{\delta} = \bar{\delta}' - \bar{\delta}$, the latency error can be given as

$$\Delta t_{\rm tot}^* = \frac{\Delta\bar{\delta} \bar{d}_{r^*} \left(1+\eta_{\rm LT}\right) m}{\sum\limits_{i=1}^{n} \frac{1-\varepsilon_i^{k_2}}{a_i^{\rm cmp}}}, \tag{42}$$

if all n workers are not stragglers. On the other hand, we denote r'^* as the merging parameter obtained by minimizing the exact total latency. Assume that the error between r^* obtained by Eq. (29) and r'^* is $\Delta r^* = r'^* - r^*$. Then, the corresponding sub-block size error caused by Δr^* can be given as

$$\Delta b_i^* = \left(\frac{1}{r^*+\Delta r^*} - \frac{1}{r^*}\right) \frac{\left(1+\eta_{\rm LT}\right) m}{k_1 a_i^{\rm cmp} \sum\limits_{i'=1}^{n} \frac{1-\varepsilon_{i'}^{k_2}}{a_{i'}^{\rm cmp}}}, i \in [n], \tag{43}$$

when all n workers are not stragglers. Moreover, the relative errors of the straggling factor caused by $\Delta\bar{\delta}$ and Δr^*, denoted as $\epsilon\lambda_{i,\Delta\bar{\delta}}''$ and $\epsilon\lambda_{i,\Delta r^*}''$, can be given as

$$\epsilon\lambda_{i,\Delta\bar{\delta}}'' = \frac{\Delta\bar{\delta} r^* \bar{d}_{r^*} \left(1-\varepsilon_i\right) \mu_i^{\rm trn}}{\lambda_{i,\bar{\delta}'}''}, \tag{44}$$

$$\epsilon\lambda_{i,\Delta r^*}'' = \bar{\delta} \left(1-\varepsilon_i\right) \mu_i^{\rm trn} \times \frac{w_1 \left(r'^* \ln \frac{m}{r'^*} - r^* \ln \frac{m}{r^*}\right) + w_2 \Delta r^*}{\lambda_{i,r'^*}''}, \tag{45}$$

for worker $i \in [n]$, where $\lambda_{i,\bar{\delta}'}''$ and λ_{i,r'^*}'' are straggling factors obtained by $\bar{\delta}'$ and r'^*, respectively.

Proof. See Appendix F. ∎

Remark 1 (*Special Cases of Merging Parameter and Sub-Block Size*). The generalization of the existing works is shown as follows:

- *When $r^* = 1$, the merged-r coded computation degenerates to the classical LT coding scheme.* Moreover, if $b_i^* = 1, i \in [n]$, the degradation leads to the fine-grained LT coding scheme [9]; if $\{b_i^*\}_{i=1}^n$ are obtained by [11, Alg. 1], the degradation leads to the block-design based wireless LT coded computation [11].
- *When $r^* = m$, the merged-r coded computation degenerates to the classical m-replication coding scheme* [2].

Remark 2 (*Trade-off between Computation, Transmission and Decoding Latency*). For merged-r LT coded computation, both r and $\{b_i\}_{i=1}^n$ affect T_{cmp} and T_{trn}, while T_{dec} is only affected by r. WNLMrLTCC realizes a trade-off between computation, transmission and decoding latency. For example, if the computation capability of each worker is the same, i.e. $\mu_i^{\mathrm{cmp}} = \mu_{\mathrm{cmp}}$ and $a_i^{\mathrm{cmp}} = a_{\mathrm{cmp}}$ for $i \in [n]$, the weaker transmission and decoding capability for the whole network will lead to the larger r^* obtained by WNLMrLTCC to reduce the transmission and decoding latency. Moreover, the workers with more powerful transmission capability and better channel condition will calculate more sub-functions. In other words, the larger μ_i^{trn} and the smaller ε_i lead to the larger kb_i^* in WNLMrLTCC for worker i.

V. Simulation Results and Discussion

In this section, we will show the performance of our proposed merged-r LT coded computation.

In order to show the superiority of the proposed scheme, a non-linear computational task in Category 3 $F(\mathbf{x}) = \mathbf{a}^{\mathrm{T}} \boldsymbol{f}(\mathbf{x})$ is considered. Similar to [5] and [8], we choose the number of non-linear sub-functions in $F(\cdot)$ as $m = 5000$, the number of workers as $n = 100$, the maximum number of sub-blocks that can be transmitted by each worker as $k_1 = 4$, and the maximum number of re-transmissions for each sub-block as $k_2 = 5$. Also, we assume the encoding parameter of LT code as $\alpha = 2$. Based on [11] and [45], the degree and decoding parameter can be determined as $w_1 = 1.3129$, $w_2 = 1.6511$ and $\eta_{\mathrm{LT}} = 0.0326$, respectively. Moreover, the decoding capability of master in both BP strategy and MDD strategy is chosen as $\mu_{\mathrm{dec}} = 100$ and $a_{\mathrm{dec}} = 0.01$[7]. The schemes studied are described as follows:

1) **UUA (Uniform Uncoded Allocation).** Each worker is assigned the same number of non-linear sub-functions without sub-block division, i.e., $b_i = m/n$, $\forall i \in [n]$;
2) **MG-LTCA (Maximum-Grained LT Coding Approach).** Each worker is assigned the same number of encoded sub-functions with maximum-grained sub-block division [9, Sec. 3.2] based on LT code, i.e., $b_i = \alpha m/k_1 n$, $\forall i \in [n]$. The BP strategy is applied;
3) **BD-WLTCC (Block-Design Based Wireless LT Coded Computation).** With the given $\{\mu_i^{\mathrm{cmp}}, a_i^{\mathrm{cmp}}, \mu_i^{\mathrm{trn}}, \varepsilon_i\}$, each worker is assigned the sub-block size b_i, $\forall i \in [n]$ based on [11, Alg. 1]. The BP strategy is applied;

[7]The unit of $1/\mu_i^{\mathrm{cmp}}$ and a_i^{cmp} is milliseconds per non-linear sub-function. The unit of μ_i^{trn} is the number of computed results for encoded sub-functions per millisecond. And the unit of $1/\mu_{\mathrm{dec}}$ and a_{dec} is milliseconds per computed result of encoded sub-function.

TABLE III
Parameters of Three Scenarios.

Scenario 1 $\delta \sim \mathcal{U}(1,7)$	Group 1: 20 workers	Group 2: 40 workers
	$\mu_i^{\mathrm{cmp}} = 300$, $a_i^{\mathrm{cmp}} = 0.02$, $\mu_i^{\mathrm{trn}} = 100$, $\varepsilon_i = 0$	$\mu_i^{\mathrm{cmp}} = 200$, $a_i^{\mathrm{cmp}} = 0.05$, $\mu_i^{\mathrm{trn}} = 100$, $\varepsilon_i = 0$
	Group 3: 30 workers	Group 4: 10 workers
	$\mu_i^{\mathrm{cmp}} = 100$, $a_i^{\mathrm{cmp}} = 0.06$, $\mu_i^{\mathrm{trn}} = 100$, $\varepsilon_i = 0$	$\mu_i^{\mathrm{cmp}} = 0.25$, $a_i^{\mathrm{cmp}} = 0.5$, $\mu_i^{\mathrm{trn}} = 100$, $\varepsilon_i = 0$
Scenario 2 $\delta \sim \mathcal{U}(1,7)$	Group 1: 20 workers	Group 2: 40 workers
	$\mu_i^{\mathrm{cmp}} = 300$, $a_i^{\mathrm{cmp}} = 0.02$, $\mu_i^{\mathrm{trn}} = 2$, $\varepsilon_i = 0.05$	$\mu_i^{\mathrm{cmp}} = 200$, $a_i^{\mathrm{cmp}} = 0.05$, $\mu_i^{\mathrm{trn}} = 10$, $\varepsilon_i = 0.15$
	Group 3: 30 workers	Group 4: 10 workers
	$\mu_i^{\mathrm{cmp}} = 100$, $a_i^{\mathrm{cmp}} = 0.06$, $\mu_i^{\mathrm{trn}} = 5$, $\varepsilon_i = 0.1$	$\mu_i^{\mathrm{cmp}} = 0.25$, $a_i^{\mathrm{cmp}} = 0.5$, $\mu_i^{\mathrm{trn}} = 1$, $\varepsilon_i = 0.2$
Scenario 3 $\delta \sim \mathcal{U}(20,30)$	100 workers	
	$\mu_i^{\mathrm{cmp}} \sim \mathcal{U}(0.1, 500)$, $a_i^{\mathrm{cmp}} \sim \mathcal{U}(0.005, 0.5)$, $\mu_i^{\mathrm{trn}} \sim \mathcal{U}(0.5, 10)$, $\varepsilon_i \sim \mathcal{U}(0, 0.5)$	

4) **MDD-WLTCC (Maximum Degree Decoding Based Wireless LT Coded Computation).** With the given $\{\mu_i^{\mathrm{cmp}}, a_i^{\mathrm{cmp}}, \mu_i^{\mathrm{trn}}, \varepsilon_i\}$, each worker is assigned the sub-block size b_i, $\forall i \in [n]$ based on [11, Alg. 1]. The MDD strategy is applied;
5) **MG-MrLTCA (Maximum-Grained Merged-r LT Coding Approach).** Each worker is assigned the same number of encoded sub-functions with maximum-grained sub-block division [9, Sec. 3.2] based on merged-r LT code, i.e., r is obtained by setting the first derivative of the upper bound of total latency to zero and $b_i = \alpha m/k_1 r n$, $\forall i \in [n]$. The BP strategy is applied;
6) **WNLMrLTCC.** The merging parameter r and sub-block size b_i, $\forall i \in [n]$ can be obtained by Alg. 2 with the given $\{\bar{\delta}, \mu_i^{\mathrm{cmp}}, a_i^{\mathrm{cmp}}, \mu_i^{\mathrm{trn}}, \varepsilon_i, \mu_{\mathrm{dec}}, a_{\mathrm{dec}}\}$. The MDD strategy is applied.

Among them, WNLMrLTCC is our proposed scheme. In order to compare the performance of different schemes, three scenarios are considered, as shown in Table III[7]. For Scenario 1, without considering transmission failure, 100 workers are divided into four groups with the different computation capabilities and the same transmission capabilities, whereas each group in Scenario 2 has disparate computation capabilities, transmission capabilities and channel conditions. In these two scenarios, the workers of Group 4 are considered as stragglers. Scenario 3 is the case of heterogeneous wireless networks where the parameters of each worker are drawn from the corresponding random sources. Besides, Scenario 1 and Scenario 2 have low-complexity task, where $\delta_\nu \sim \mathcal{U}(1,7), \nu \in [m]$, while the complexity of desired computational task in Scenario 3 is much higher as $\delta_\nu \sim \mathcal{U}(20,30), \nu \in [m]$.

Performance comparisons in the above three scenarios between the implemented schemes are shown in Fig. 7. It can be observed that WNLMrLTCC can achieve an optimal trade-off between computation, transmission and decoding latency and reduce the total latency as much as possible due to the more efficient coding and decoding strategy for non-linear tasks compared with other schemes. Furthermore, the upper bound of total latency changes over r are simulated in Fig. 8 within these three scenarios. From Fig. 8, there is a value of r that makes the upper bound achieve the minimum, and

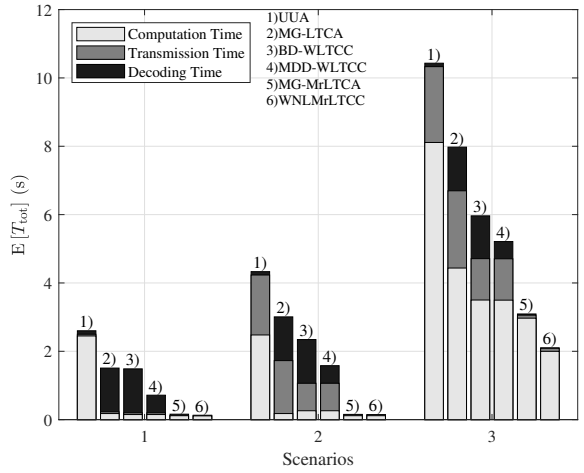


Fig. 7. Latency comparisons between WNLMrLTCC and five benchmarks in three different scenarios.

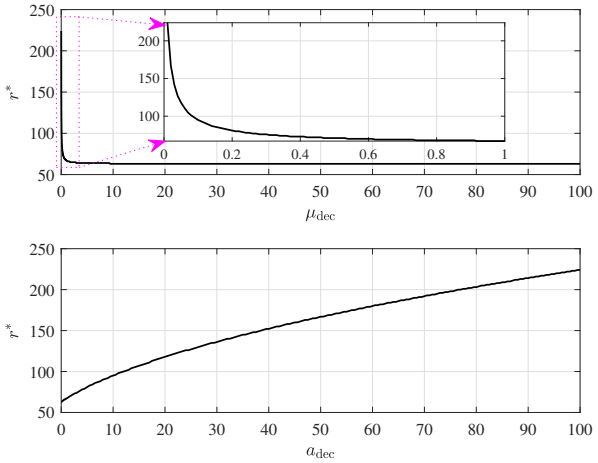


Fig. 9. The optimal r^* chosen in WNLMrLTCC versus the decoding capability of master $\{\mu_{\rm dec}, a_{\rm dec}\}$, where the parameters of workers are chosen from Scenario 2.

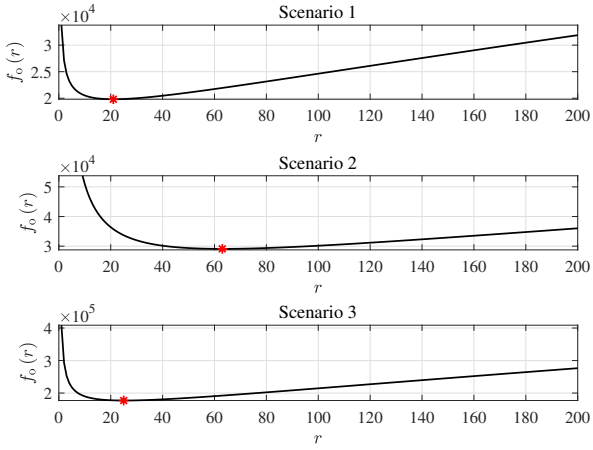


Fig. 8. The upper bound of total latency $f_{\rm o}(r)$ versus r, where the marker '*' represents the optimal r^* chosen in WNLMrLTCC.

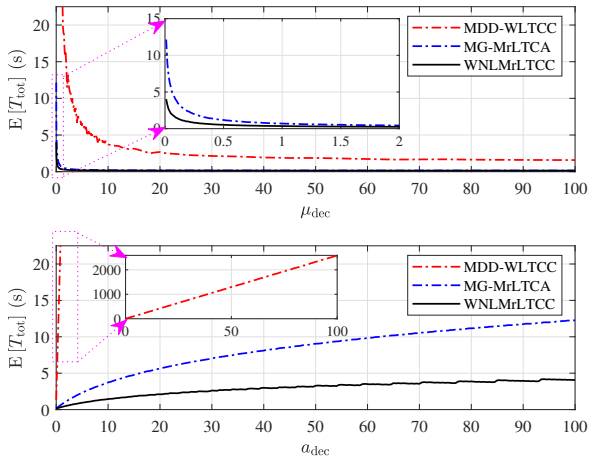


Fig. 10. The total expected latency $\mathbb{E}[T_{\rm tot}]$ versus the decoding capability of master $\{\mu_{\rm dec}, a_{\rm dec}\}$, where the parameters of workers are chosen from Scenario 2.

WNLMrLTCC can always obtain this optimal r^*.

For Scenario 2, the changes of r^* and $\mathbb{E}[T_{\rm tot}]$ over $\{\mu_{\rm dec}, a_{\rm dec}\}$ are shown in Fig. 9 and Fig. 10. We can observe that the total latency will increase as the decoding capability of master becomes weaker because of the increase in the decoding latency. At this moment, the performance of the whole network is straggled by the poor decoding capability. In order to balance the computation, transmission and decoding latency, the merging parameter r^* in WNLMrLTCC will increase. This reduces the decoding threshold and also transmission overhead, which leads to a much lower total latency. Especially, a smaller $\mu_{\rm dec}$ or a larger $a_{\rm dec}$ can result in a higher performance gain compared with other schemes.

Fig. 11 and Fig. 12 show the effect of $\bar{\delta}$ on r^* and $\mathbb{E}[T_{\rm tot}]$ respectively. We can note that the computation latency will increase when $\bar{\delta}$ increases, which may lead to severe latency performance degradation caused by computation stragglers. At this moment, r^* in WNLMrLTCC will decrease to balance the total latency so that each sub-block consists of fewer non-linear sub-functions. Then, the computation latency will also be reduced to achieve better performance. In particular, more significant latency performance improvements will be achieved by WNLMrLTCC as $\bar{\delta}$ becomes larger.

VI. Conclusion

In this paper, we have proposed the merged-r LT coded computation scheme for non-linear tasks. First, in order to reduce the high computation and transmission costs, the merged-r LT coding strategy has been presented. Then, the efficient MDD strategy has been given without recovering the individual non-linear sub-functions. We have also shown the advantages of the proposed strategies from a theoretical perspective. Finally, to complete these heterogeneous sub-tasks with low latency, we have analyzed and optimized the total latency including computation, transmission and decoding

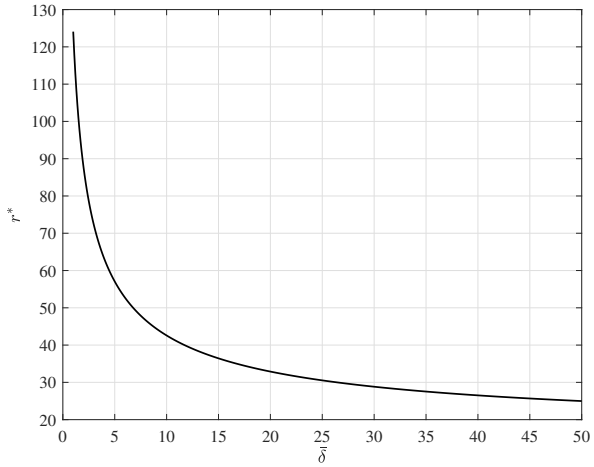


Fig. 11. The optimal r^* chosen in WNLMrLTCC versus the average computation complexity of non-linear sub-functions $\bar{\delta}$, where the parameters of workers are chosen from Scenario 2.

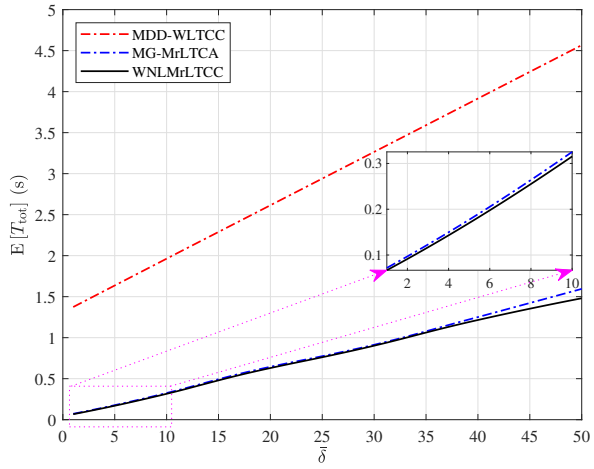


Fig. 12. The total expected latency $\mathbb{E}[T_{\text{tot}}]$ versus the average computation complexity of non-linear sub-functions $\bar{\delta}$, where the parameters of workers are chosen from Scenario 2.

process. The WNLMrLTCC algorithm has been presented to obtain the optimal merging parameter and sub-block size. The simulation results have also verified the superiority of our proposed scheme.

Appendix A
Proof of Corollary 1

For BP strategy, only a single merged sub-function can be recovered in each iteration from the worst point of view. At this point, the degree of the encoded sub-functions presents a characteristic of constant difference. To recover m/r merged sub-functions, master needs a total of $(m/2r)(m/r-1)$ additions. At last, master sums these merged sub-functions up to obtain the desired result, which needs $m/r-1$ additions. Then,

$$N_{\text{dec,b}}^{\text{BP}} = \left(\frac{m}{2r}+1\right)\left(\frac{m}{r}-1\right).$$

For MDD strategy, from the worst point of view, there is no encoded sub-function disconnected with the base function, and the maximum degree in received results can be identified as $d_{\max} = \bar{d}_r$. In other words, there are a total of $m/r - \bar{d}_r$ merged sub-functions that need to be recovered through iterative peeling algorithm. Similarly, master needs at most $\left(m/2r - \bar{d}_r/2\right)\left(m/r - \bar{d}_r + 1\right)$ additions to decode these merged sub-functions successfully in this case. Then, the sum of these $m/r - \bar{d}_r$ merged sub-functions and base function is obtained to recover the desired result, which yields

$$N_{\text{dec,b}}^{\text{MDD}} = \frac{1}{2}\left(\frac{m}{r}-\bar{d}_r\right)\left(\frac{m}{r}-\bar{d}_r+3\right).$$

Thus, Eq. (10) can be derived as

$$\begin{aligned} N_{\text{dec,b}}^{\text{BP}} - N_{\text{dec,b}}^{\text{MDD}} =& \frac{m}{r}\bar{d}_r + \frac{3}{2}\bar{d}_r - \frac{1}{2}\bar{d}_r^2 - \frac{m}{r} - 1 \\ =& \Theta\left(\frac{m}{r}\ln\frac{m}{r}\right). \end{aligned}$$

Appendix B
Proof of Proposition 2

When the computation capability tuple for all n workers is $\{\mu_{\text{b}}^{\text{cmp}}, a_{\text{b}}^{\text{cmp}}\}$, the computation performance of the whole network achieves the worst case. It implies that the heterogeneous networks are reduced to the corresponding homogeneous networks consisting of n workers with the same computation capability $\{\mu_{\text{b}}^{\text{cmp}}, a_{\text{b}}^{\text{cmp}}\}$. At this point, we can know that $T_i^{\text{cmp}} = c_i b_i \bar{\delta} r \bar{d}_r (\hat{T}_{i,\text{b}}^{\text{cmp}} + a_{\text{b}}^{\text{cmp}})$ for $i \in [n]$, where the random variable $\hat{T}_{i,\text{b}}^{\text{cmp}}$ is exponentially distributed with rate parameter $\mu_{\text{b}}^{\text{cmp}}$. Then,

$$T_{\text{cmp}} \le (c_{i'}+1)\, b_{i'} \bar{\delta} r \bar{d}_r \left(\hat{T}_{i',\text{b}}^{\text{cmp}} + a_{\text{b}}^{\text{cmp}}\right), i' \in \mathcal{W}_{\text{cs}},$$

where $\mathcal{W}_{\text{cs}}$ represents the set of workers that have not completed all their computational sub-tasks until T_{cmp}, i.e. $\mathcal{W}_{\text{cs}} = \{i \,|\, c_i b_i < \alpha m/rn, c_i < k_1, i \in [n]\}$. Summing over all $i' \in \mathcal{W}_{\text{cs}}$, we can get

$$\begin{aligned} \sum_{i' \in \mathcal{W}_{\text{cs}}} T_{\text{cmp}} &\le \sum_{i' \in \mathcal{W}_{\text{cs}}} (c_{i'}+1)\, b_{i'} \bar{\delta} r \bar{d}_r \left(\hat{T}_{i',\text{b}}^{\text{cmp}} + a_{\text{b}}^{\text{cmp}}\right) \\ &\le \sum_{i' \in \mathcal{W}_{\text{cs}}} \bar{\delta} r \bar{d}_r \left(\hat{T}_{i',\text{b}}^{\text{cmp}} + a_{\text{b}}^{\text{cmp}}\right) \max_{i' \in \mathcal{W}_{\text{cs}}} (c_{i'}+1)\, b_{i'} \\ &\le \left(\frac{\alpha m}{rn}+1\right) \sum_{i' \in \mathcal{W}_{\text{cs}}} \bar{\delta} r \bar{d}_r \left(\hat{T}_{i',\text{b}}^{\text{cmp}} + a_{\text{b}}^{\text{cmp}}\right), \\ \Rightarrow \quad T_{\text{cmp}} &\le \left(\frac{\alpha m}{rn}+1\right) \bar{\delta} r \bar{d}_r \left(\bar{\hat{T}}_{\text{b}}^{\text{cmp}} + a_{\text{b}}^{\text{cmp}}\right), \end{aligned}$$

where $\bar{\hat{T}}_{\text{b}}^{\text{cmp}} = \sum_{i'} \hat{T}_{i',\text{b}}^{\text{cmp}} / \sum_{i'} 1, i' \in W_{\text{cs}}$. Thus, the expected upper bound (18) can be obtained.

On the other hand, when the computation capability tuple for all n workers is $\left\{\mu_{\text{g}}^{\text{cmp}}, a_{\text{g}}^{\text{cmp}}\right\}$, the heterogeneous networks are reduced to the corresponding homogeneous networks with the best computation performance. It implies that

$T_i^{\rm cmp} = c_i b_i \bar{\delta} r \bar{d}_r (\hat{T}_{i,\rm g}^{\rm cmp} + a_{\rm g}^{\rm cmp})$ for $i \in [n]$, where $\hat{T}_{i,\rm g}^{\rm cmp}$ is an exponential random variable with rate parameter $\mu_{\rm g}^{\rm cmp}$. Then,

$$\begin{aligned} T_{\rm cmp} &= \max_{i\in[n]} T_i^{\rm cmp} = \max_{i\in[n]} c_i b_i \bar{\delta} r \bar{d}_r (\hat{T}_{i,\rm g}^{\rm cmp} + a_{\rm g}^{\rm cmp}) \\ &\geq \max_{i\in[n]} c_i b_i \bar{\delta} r \bar{d}_r \min_{i\in[n]} (\hat{T}_{i,\rm g}^{\rm cmp} + a_{\rm g}^{\rm cmp}) \\ &\geq \frac{(1+\eta_{\rm LT})m}{(1-\varepsilon_{\rm g}^{k_2})n} \bar{\delta} \bar{d}_r (\hat{T}_{(1)}^{\rm cmp} + a_{\rm g}^{\rm cmp}), \end{aligned}$$

where $\hat{T}_{(1)}^{\rm cmp}$ is the first order statistic of $\{\hat{T}_{i,\rm g}^{\rm cmp}\}_{i=1}^n$. Thus, the expected lower bound (19) can be obtained.

Appendix C
Proof of Proposition 3

Using Eq. (23) and the definition of the mean, we obtain

$$\begin{aligned} \mathbb{E}\left[K_i^{\rm trn}\right] &= \sum_{\kappa=1}^{k_2-1} \kappa \varepsilon_i^{\kappa-1}(1-\varepsilon_i) + k_2 \varepsilon_i^{k_2-1} \\ &= (1-\varepsilon_i)\frac{\partial}{\partial \varepsilon_i}\Big(\sum_{\kappa=1}^{k_2-1} \varepsilon_i^{\kappa}\Big) + k_2 \varepsilon_i^{k_2-1} = \frac{1-\varepsilon_i^{k_2}}{1-\varepsilon_i}. \end{aligned}$$

Then, based on Eq. (22) and Wald's lemma [46], we can get

$$\begin{aligned} \mathbb{E}\left[T_{i,j'}^{\rm trn}\right] &= \mathbb{E}\Big[\sum_{\kappa=1}^{K_i^{\rm trn}} T_{i,j',\kappa}^{\rm trn}\Big] = \mathbb{E}\left[K_i^{\rm trn}\right]\mathbb{E}\left[T_{i,j',\kappa}^{\rm trn}\right] \\ &= \frac{1-\varepsilon_i^{k_2}}{(1-\varepsilon_i)\mu_i^{\rm trn}}. \end{aligned}$$

Similarly, $\mathbb{E}\left[T_i^{\rm trn}\right]$ can be given as

$$\mathbb{E}\left[T_i^{\rm trn}\right] = \mathbb{E}\Big[\sum_{j'=1}^{c_i b_i} T_{i,j'}^{\rm trn}\Big] = b_i \mathbb{E}\left[c_i\right]\mathbb{E}\left[T_{i,j'}^{\rm trn}\right].$$

Thus, Eq. (24) can be obtained.

Appendix D
Proof of Proposition 5

Based on Eq. (11), Eq. (14), Proposition 2, Proposition 3 and Proposition 4, the upper bound of $\mathbb{E}\left[T_{\rm tot}\right]$ can be derived as

$$\begin{aligned} \mathbb{E}[T_{\rm tot}] &= \mathbb{E}\left[T_{\rm cmp}\right] + \sum_{i=1}^{n} \mathbb{E}\left[c_i b_i\right]\mathbb{E}\left[T_{i,j'}^{\rm trn}\right] + \mathbb{E}\left[T_{\rm dec}\right] \\ &\leq \mathbb{E}\left[T_{\rm cmp}\right] + \sum_{i=1}^{n} \frac{\mathbb{E}\left[T_{\rm cmp}\right]\mathbb{E}\left[T_{i,j'}^{\rm trn}\right]}{\mathbb{E}[\bar{\delta} r \bar{d}_r (\hat{T}_i^{\rm cmp} + a_i^{\rm cmp})]} + \mathbb{E}\left[T_{\rm dec}\right] \\ &\leq \bar{\delta}\big(\frac{\alpha m}{n} + r\big)\bar{d}_r s_{\rm b}^{\rm cmp}\Big(1 + \sum_{i=1}^{n} \frac{1-\varepsilon_i^{k_2}}{\bar{\delta} r \bar{d}_r s_i^{\rm cmp}(1-\varepsilon_i)\mu_i^{\rm trn}}\Big) \\ &\quad + s_{\rm dec}\big(\frac{m}{r} - \bar{d}_r\big)\bar{d}_r = f_{\rm o}(r). \end{aligned}$$

Because $f_{\rm o}(r), r \in [m]$ is a continuous function, the minimum of the upper bound can be obtained when $r = 1$, $r = \tilde{r}$ or $r = m$, which yields Eq. (29).

Appendix E
Proof of Corollary 2

When $\mu_i^{\rm cmp} \cdot a_i^{\rm cmp}$ is large, based on Euler-Maclaurin formula, Eq. (36), Eq. (39) and Eq. (37) can be rewritten as

$$\begin{aligned} h_i &= k_1 - e^{\mu_i^{\rm cmp} a_i^{\rm cmp}}\big(k_1 e^{-\frac{\mu_i^{\rm cmp}\gamma_i}{k_1 \bar{\delta} r \bar{d}_r}} - e^{-\frac{\mu_i^{\rm cmp}\gamma_i}{\bar{\delta} r \bar{d}_r}} - \frac{\mu_i^{\rm cmp}\gamma_i}{\bar{\delta} r \bar{d}_r}\tilde{E}_i\big), \\ g_i &= (1-\varepsilon_i^{k_2}) e^{\mu_i^{\rm cmp} a_i^{\rm cmp}} \frac{\mu_i^{\rm cmp}}{\bar{\delta} r \bar{d}_r}\tilde{E}_i, \\ &- k_1 + e^{\mu_i^{\rm cmp} a_i^{\rm cmp}}\big(k_1 e^{-\frac{\mu_i^{\rm cmp}\gamma_i}{k_1 \bar{\delta} r \bar{d}_r}} - e^{-\frac{\mu_i^{\rm cmp}\gamma_i}{\bar{\delta} r \bar{d}_r}}\big) = 0, \end{aligned}$$

where $\tilde{E}_i = \mathrm{Ei}_1(\mu_i^{\rm cmp}\gamma_i / k_1 \bar{\delta} r \bar{d}_r) - \mathrm{Ei}_1(\mu_i^{\rm cmp}\gamma_i / \bar{\delta} r \bar{d}_r)$, and $\mathrm{Ei}_1(x) = \int_x^\infty e^{-y}/y \mathrm{d}y$ is the exponential integral. As $\mu_i^{\rm cmp} \cdot a_i^{\rm cmp}$ becomes large, we notice $e^{-(k_1-1)\mu_i^{\rm cmp} a_i^{\rm cmp}} \to 0$. Then, the solution to Eq. (37) and $\tilde{E}_i$ are reduced to

$$\gamma_i = k_1 \bar{\delta} r \bar{d}_r a_i^{\rm cmp}, \ \tilde{E}_i = \frac{e^{-\mu_i^{\rm cmp} a_i^{\rm cmp}}}{\mu_i^{\rm cmp} a_i^{\rm cmp}} - \frac{e^{-k_1 \mu_i^{\rm cmp} a_i^{\rm cmp}}}{k_1 \mu_i^{\rm cmp} a_i^{\rm cmp}}.$$

Thus, we can get $h_i = k_1$ and $g_i = (1-\varepsilon_i^{k_2})\Big/\bar{\delta} r \bar{d}_r a_i^{\rm cmp}$, respectively. According to Eq. (34), Eq. (38) and Eq. (40), set $\lambda_i'' = \bar{\delta} r \bar{d}_r \lambda_i'$, and we can obtain Eq. (41) and

$$b_i^* = \frac{t_{\rm cmp}^*}{\gamma_i} = \frac{(1+\eta_{\rm LT})m}{k_1 r a_i^{\rm cmp} \sum_{i'=1}^{n} \frac{1-\varepsilon_{i'}^{k_2}}{a_{i'}^{\rm cmp}}} \to 1,$$

since k_1 and l are large. It implies that the sub-block size obtained by WNLMrLTCC should be small when the transmission rounds are unlimited. Moreover, as $k_2 \to \infty$, we can observe $p_{{\rm f},i} = \varepsilon_i^{k_2} \to 0$. At this point, η is only decided by the LT decoding process. It means that η is reduced to $\eta_{\rm LT}$.

Appendix F
Proof of Corollary 3

According to Appendix E, the optimal time and sub-block size, and the corresponding straggling factor can be given as

$$\begin{aligned} t_{\rm cmp}^* &= \frac{\bar{\delta}\bar{d}_{r^*}(1+\eta_{\rm LT})m}{\sum_{i=1}^{n} \frac{1-\varepsilon_i^{k_2}}{a_i^{\rm cmp}} \cdot \mathbb{1}_{\{\lambda_i''>0\}}}, \\ t_i^{\rm trn*} &= \frac{t_{\rm cmp}^*\big(1-\varepsilon_i^{k_2}\big)}{\bar{\delta} r^* \bar{d}_{r^*} a_i^{\rm cmp}(1-\varepsilon_i)\mu_i^{\rm trn}} \cdot \mathbb{1}_{\{\lambda_i''>0\}}, i \in [n], \\ b_i^* &= \frac{(1+\eta_{\rm LT})m}{k_1 r^* a_i^{\rm cmp} \sum_{i'=1}^{n} \frac{1-\varepsilon_{i'}^{k_2}}{a_{i'}^{\rm cmp}} \cdot \mathbb{1}_{\{\lambda_{i'}''>0\}}} \cdot \mathbb{1}_{\{\lambda_i''>0\}}, i \in [n], \\ \lambda_i'' &= \bar{\delta} r^* \bar{d}_{r^*}(1-\varepsilon_i)\mu_i^{\rm trn} - \sum_{i'=1}^{n} \frac{1-\varepsilon_{i'}^{k_2}}{a_{i'}^{\rm cmp}} \\ &\quad + (1-\varepsilon_i)\mu_i^{\rm trn} \sum_{i'=1}^{n} \frac{1-\varepsilon_{i'}^{k_2}}{(1-\varepsilon_{i'})\mu_{i'}^{\rm trn} a_{i'}^{\rm cmp}}, i \in [n], \end{aligned}$$

when $\mu_i^{\rm cmp}\cdot a_i^{\rm cmp}$ is large. Since $\bar{\delta}$ does not affect the decoding latency, when all n workers are not stragglers, the latency error can be given as

$$\begin{aligned}\Delta t_{\rm tot}^* =& t_{\rm cmp}'^* + \sum_{i=1}^{n} t_i'^{\rm trn*} - \left(t_{\rm cmp}^* + \sum_{i=1}^{n} t_i^{\rm trn*}\right)\\ =& t_{\rm cmp}'^* - t_{\rm cmp}^*\\ &+ \left(\frac{t_{\rm cmp}'^*}{\bar{\delta}+\Delta\bar{\delta}} - \frac{t_{\rm cmp}^*}{\bar{\delta}}\right)\sum_{i=1}^{n}\frac{1-\varepsilon_i^{k_2}}{r^*\bar{d}_{r^*}a_i^{\rm cmp}\left(1-\varepsilon_i\right)\mu_i^{\rm trn}}\\ =& t_{\rm cmp}'^* - t_{\rm cmp}^* = \frac{\Delta\bar{\delta}\bar{d}_{r^*}\left(1+\eta_{\rm LT}\right)m}{\sum\limits_{i=1}^{n}\frac{1-\varepsilon_i^{k_2}}{a_i^{\rm cmp}}},\end{aligned}$$

where $t_{\rm cmp}'^*$ and $t_i'^{\rm trn*}$ are the computation time and the transmission time obtained by $\bar{\delta}'$, respectively. This implies $\Delta\bar{\delta}$ only affects the computation latency. Besides, the relative error of the straggling factor caused by $\Delta\bar{\delta}$ is derived as

$$\epsilon\lambda''_{i,\Delta\bar{\delta}} = \frac{\lambda''_{i,\bar{\delta}'} - \lambda''_i}{\lambda''_{i,\bar{\delta}'}} = \frac{\Delta\bar{\delta}r^*\bar{d}_{r^*}\left(1-\varepsilon_i\right)\mu_i^{\rm trn}}{\lambda''_{i,\bar{\delta}'}}.$$

Similarly, the sub-block size error caused by Δr^* can be given as

$$\begin{aligned}\Delta b_i^* =& b_i'^* - b_i^*\\ =& \frac{\left(1+\eta_{\rm LT}\right)m}{k_1\left(r^*+\Delta r^*\right)a_i^{\rm cmp}\sum\limits_{i'=1}^{n}\frac{1-\varepsilon_{i'}^{k_2}}{a_{i'}^{\rm cmp}}} - \frac{\left(1+\eta_{\rm LT}\right)m}{k_1 r^* a_i^{\rm cmp}\sum\limits_{i'=1}^{n}\frac{1-\varepsilon_{i'}^{k_2}}{a_{i'}^{\rm cmp}}}\\ =& \left(\frac{1}{r^*+\Delta r^*} - \frac{1}{r^*}\right)\frac{\left(1+\eta_{\rm LT}\right)m}{k_1 a_i^{\rm cmp}\sum\limits_{i'=1}^{n}\frac{1-\varepsilon_{i'}^{k_2}}{a_{i'}^{\rm cmp}}},\end{aligned}$$

where $b_i'^*$ is the sub-block size obtained by r'^*. Moreover, the relative error of the straggling factor caused by Δr^* is derived as

$$\begin{aligned}\epsilon\lambda''_{i,\Delta r^*} =& \frac{\lambda''_{i,r'^*} - \lambda''_i}{\lambda''_{i,r'^*}} = \bar{\delta}\left(1-\varepsilon_i\right)\mu_i^{\rm trn}\frac{r'^*\bar{d}_{r'^*} - r^*\bar{d}_{r^*}}{\lambda''_{i,r'^*}}\\ =& \bar{\delta}\left(1-\varepsilon_i\right)\mu_i^{\rm trn}\frac{w_1\left(r'^*\ln\frac{m}{r'^*} - r^*\ln\frac{m}{r^*}\right) + w_2\Delta r^*}{\lambda''_{i,r'^*}}.\end{aligned}$$

REFERENCES

[1] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Proces. Syst. (NIPS)*, vol. 25, 2012, pp. 1223–1231.

[2] D. Wang, G. Joshi, and G. W. Wornell, "Efficient straggler replication in large-scale parallel computing," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 4, no. 2, apr 2019.

[3] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[4] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theor. (ISIT)*, 2017, pp. 2418–2422.

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

[6] D.-J. Han, J.-Y. Sohn, and J. Moon, "Coded wireless distributed computing with packet losses and retransmissions," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 8204–8217, 2021.

[7] F. Wu and L. Chen, "Latency optimization for coded computation straggled by wireless transmission," *IEEE Wireless Commun. Lett.*, vol. 9, no. 7, pp. 1124–1128, 2020.

[8] L. Chen, K. Han, Y. Du, and Z. Wang, "Block-division-based wireless coded computation," *IEEE Wireless Commun. Lett.*, vol. 11, no. 2, pp. 283–287, 2022.

[9] A. Mallick, M. Chaudhari, U. Sheth, G. Palanikumar, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, dec 2019.

[10] X. Yang, M. Jiang, and C. Zhao, "LT codes with feedback: Accelerate the distributed matrix-vector multiplication with stragglers," in *Proc. IEEE Int. Perform. Comput. Commun. Conf. (IPCCC)*, 2019, pp. 1–6.

[11] B. Fang, K. Han, Z. Wang, and L. Chen, "Latency optimization for luby transform coded computation in wireless networks," *IEEE Wireless Commun. Lett.*, vol. 12, no. 2, pp. 197–201, 2023.

[12] B. Fang, L. Chen, Y. Chen, C. You, X. Chen, and W. Wang, "Wireless coded computation with error detection," *IEEE Trans. Commun.*, vol. 72, no. 3, pp. 1273–1289, 2024.

[13] A. B. Das and A. Ramamoorthy, "Coded sparse matrix computation schemes that leverage partial stragglers," *IEEE Trans. Inf. Theory*, vol. 68, no. 6, pp. 4156–4181, 2022.

[14] J.-A. Lin, Y.-C. Huang, M.-C. Lee, and P.-N. Chen, "Coded distributed multiplication for matrices of different sparsity levels," *IEEE Trans. Commun.*, vol. 72, no. 2, pp. 633–647, 2024.

[15] R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Adaptive private distributed matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 68, no. 4, pp. 2653–2673, 2022.

[16] S. Hong, H. Yang, and J. Lee, "Hierarchical group testing for byzantine attack identification in distributed matrix multiplication," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 3, pp. 1013–1029, 2022.

[17] C. Hofmeister, R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Secure private and adaptive matrix multiplication beyond the singleton bound," *IEEE J. Sel. Area. Inf. Theory*, vol. 3, no. 2, pp. 275–285, 2022.

[18] T. Jahani-Nezhad and M. A. Maddah-Ali, "Codedsketch: A coding scheme for distributed computation of approximated matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 4185–4196, 2021.

[19] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, 2018.

[20] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Timely coded computing," in *Proc. IEEE Int. Symp. Inf. Theor. (ISIT)*, 2019, pp. 2798–2802.

[21] H. Zhu, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Hierarchical coded matrix multiplication in heterogeneous multihop networks," *IEEE Trans. Commun.*, vol. 70, no. 6, pp. 3597–3612, 2022.

[22] Y. Sun, F. Zhang, J. Zhao, S. Zhou, Z. Niu, and D. Gündüz, "Coded computation across shared heterogeneous workers with communication delay," *IEEE Trans. Signal Process.*, vol. 70, pp. 3371–3385, 2022.

[23] S. S. Arslan, "Array BP-XOR codes for hierarchically distributed matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 68, no. 3, pp. 2050–2066, 2022.

[24] R. Schlegel, S. Kumar, E. Rosnes, and A. Graell i Amat, "Privacy-preserving coded mobile edge computing for low-latency distributed inference," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 3, pp. 788–799, 2022.

[25] D. Ko, S. H. Chae, and W. Choi, "MDS coded task offloading in stochastic wireless edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 2107–2121, 2022.

[26] B. Pang, S. Gu, Q. Zhang, N. Zhang, and W. Xiang, "CCOS: A coded computation offloading strategy for satellite-terrestrial integrated networks," in *Proc. Int. Wirel. Commun. Mob. Comput. (IWCMC)*, 2021, pp. 242–247.

[27] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, vol. 89, 2019, pp. 1215–1225.

[28] A. Asheralieva and D. Niyato, "Throughput-efficient lagrange coded private blockchain for secured IoT systems," *IEEE Internet Things J.*, vol. 8, no. 19, pp. 14 874–14 895, 2021.

[29] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 7, 2017, pp. 5166–5178.

[30] R. Schlegel, S. Kumar, E. Rosnes, and A. G. i. Amat, "CodedPaddedFL and CodedSecAgg: Straggler mitigation and secure aggregation in federated learning," *IEEE Trans. Commun.*, vol. 71, no. 4, pp. 2013–2027, 2023.

[31] E. Ozfatura, S. Ulukus, and D. Gündüz, "Coded distributed computing with partial recovery," *IEEE Trans. Inf. Theory*, vol. 68, no. 3, pp. 1945–1959, 2022.

[32] K. Wan, H. Sun, M. Ji, and G. Caire, "Distributed linearly separable computation," *IEEE Trans. Inf. Theory*, vol. 68, no. 2, pp. 1259–1278, 2022.

[33] A. Khalesi and P. Elia, "Multi-user linearly-separable distributed computing," *IEEE Trans. Inf. Theory*, vol. 69, no. 10, pp. 6314–6339, 2023.

[34] A. Mallick, S. Smith, and G. Joshi, "Rateless codes for distributed non-linear computations," in *Proc. Int. Symp. Top. Coding (ISTC)*, 2021, pp. 1–5.

[35] A. Mallick and G. Joshi, "Rateless sum-recovery codes for distributed non-linear computations," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2022, pp. 160–165.

[36] C. A. Navarro, N. Hitschfeld-Kahler, and L. Mateu, "A survey on parallel computing and its applications in data-parallel problems using GPU architectures," *Commun. Comput. Phys.*, vol. 15, no. 2, p. 285–329, 2014.

[37] M. M. Lau and K. Hann Lim, "Review of adaptive activation function in deep neural network," in *Proc. IEEE EMBS Conf. Biomed. Eng. Sci. (IECBES)*, 2018, pp. 686–690.

[38] S. Xiong and Z. Hou, "Model-free adaptive control for unknown MIMO nonaffine nonlinear discrete-time systems with experimental validation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1727–1739, 2022.

[39] T. Song, H. Li, F. Meng, Q. Wu, and J. Cai, "LETRIST: Locally encoded transform feature histogram for rotation-invariant texture classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 7, pp. 1565–1579, 2018.

[40] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[41] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *Ann. Statist.*, vol. 36, no. 3, pp. 1171 – 1220, 2008.

[42] M. Luby, "LT codes," in *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, 2002, pp. 271–280.

[43] F. Lázaro, G. Liva, and G. Bauch, "Inactivation decoding of LT and raptor codes: Analysis and code design," *IEEE Trans. Commun.*, vol. 65, no. 10, pp. 4114–4127, 2017.

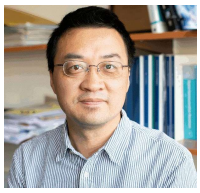
[44] V. Bioglio, "MRB decoding of LT codes over AWGN channels," *IEEE Wireless Commun. Lett.*, vol. 8, no. 2, pp. 548–551, 2019.

[45] W. Yao, B. Yi, T. Huang, and W. Li, "Poisson robust soliton distribution for LT codes," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1499–1502, 2016.

[46] J. Janssen and R. Manca, *Renewal Theory*. Boston, MA: Springer US, 2006, pp. 45–104.

Borui Fang received the B.E. degree in communication engineering from Dalian Maritime University, Dalian, China, in 2020. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China. His research interests include coded distributed computing, wireless networks, and integrated communication and computation. He served as the TPC member for IEEE VTC2022-Fall, IEEE VTC2023-Spring, 2024 IEEE Globecom Workshops, and IEEE MCSoC 2024.

Li Chen (Senior Member, IEEE) received the B.E. degree in electrical and information engineering from the Harbin Institute of Technology, Harbin, China, in 2009, and the Ph.D. degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 2014. He is currently an Associate Professor with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. His research interests include integrated communication and computation, integrated sensing and communication, and wireless IoT networks.



Yunfei Chen (Senior Member, IEEE) received his B.E. and M.E. degrees in electronics engineering from Shanghai Jiaotong University, Shanghai, China, in 1998 and 2001, respectively, and the Ph.D. degree from the University of Alberta in 2006. He is currently a Professor with the Department of Engineering, Durham University, U.K. His research interests include wireless communications, performance analysis, and joint radar communications designs.

Weidong Wang received the B.S. degree from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 1989, and the M.S. degree from the University of Science and Technology of China, Hefei, China, in 1993. He is currently a Full Professor with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. His research interests include wireless communication, microwave and millimeter-wave, and radar technology. He is a member of the Committee of Optoelectronic Technology, Chinese Society of Astronautics.