Stochastic Resource Optimization for Wireless Powered Hybrid Coded Edge Computing Networks

Wei Chong Ng, Wei Yang Bryan Lim, Zehui Xiong, Dusit Niyato, *Fellow, IEEE*, H. Vincent Poor, *Life Fellow, IEEE*, Xuemin Sherman Shen, *Fellow, IEEE*, and Chunyan Miao

***Abstract*—To enable ubiquitous Artificial Intelligence (AI) in the next-generation wireless communications networks, computation-intensive tasks such as data processing and model training have to be performed by energy-constrained end users. In this paper, we present a hybrid coded edge computing network whereby users can choose to complete their computation task through: i) local computation with the wireless power transfer derived from base stations, ii) coded edge offloading, or iii) hybrid computation involving edge offloading and local computation. To minimize the overall network cost, we propose a stochastic resource optimization approach. Given the stochastic nature of wireless charging efficiency and edge servers computation capacities, which can only be observed *ex-post*, a computation strategy for each user is determined using the two-stage stochastic integer programming (SIP). To address the complexity of the SIP problem which scales with the size of the network, we introduce the efficient computation methods of Benders' decomposition and sample average approximation. Besides, we present a special case of z-stage stochastic offloading optimization that is applicable when the corrective edge offloading action can be executed in multiple stages, e.g., for non-time-sensitive tasks that do not need to be completed by stage two. Finally, we provide extensive sensitivity analyses to evaluate the performance of the proposed cost minimization approach amid varying network parameters. We demonstrate that our approach outperforms deterministic optimization approaches for in-network cost minimization.**

***Index Terms*—Edge computing, resource management, network economics, multi-tier computing, stochastic integer programming.**

Manuscript received 28 July 2022; revised 30 December 2022; accepted 15 February 2023. Date of publication 22 February 2023; date of current version 5 February 2024. This research was supported in part by the National Research Foundation (NRF), Singapore and Infocomm Media Development Authority under the Future Communications Research Development Programme (FCP), and DSO National Laboratories under the AI Singapore Programme AISG under Grant AISG2-RP-2020-019, under Energy Research Test-Bed and Industry Partnership Funding Initiative, part of the Energy Grid (EG) 2.0 programme, under DesCartes and the Campus for Research Excellence and Technological Enterprise (CREATE) programme, Alibaba Group through Alibaba Innovative Research (AIR) Program, Alibaba-NTU Singapore Joint Research Institute (JRI), in part by the SUTD SRG-ISTD-2021-165, the SUTD-ZJU IDEA under Grant SUTD-ZJU (VP) 202102, and in part by the Ministry of Education, Singapore, under its SUTD Kickstarter Initiative under Grant SKI 20210204. The research was also supported in part by U.S National Science Foundation under Grants CCF-1908308 and CNS-2128448. Recommended for acceptance by Dr. H. Seferoglu. *(Corresponding author: Zehui Xiong.)*

Wei Chong Ng and Wei Yang Bryan Lim are with Alibaba Group and Alibaba-NTU Joint Research Institute, Nanyang Technological University (NTU), Singapore 639798 (e-mail: weichong001@e.ntu.edu.sg; limw0201@e.ntu.edu.sg).

Zehui Xiong is with the Pillar of Information Systems Technology and Design, Singapore University of Technology Design, Singapore 487372 (e-mail: zehui_xiong@sutd.edu.sg).

Dusit Niyato and Chunyan Miao are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: dniyato@ntu.edu.sg; ascymiao@ntu.edu.sg).

H. Vincent Poor is with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: poor@princeton.edu).

Xuemin Sherman Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TMC.2023.3246994

I. Introduction

TODAY, the Internet-of-Things (IoT) has become an integral aspect of our lives, disrupting the way in which we track our personal health [1], maintain manufacturing equipment [2], and monitor the environment [3]. The next-generation communication systems are expected to further promote the growth of the *Internet of Everything* paradigm [4], by leveraging the enormous amounts of sensing data to drive the development of ubiquitous intelligence using Artificial Intelligence (AI) techniques. However, the data-driven AI models rely heavily on massive quantities of training data, which in turn requires significant computation power for processing and model training. Moreover, due to the energy constraints, IoT devices face challenges in meeting the required quality-of-service (QoS) demands of the end-users, especially for latency-sensitive applications.

In response, edge computing is expected to be a key supporting pillar of next-generation communication systems. Specifically, edge computing leverages the storage, communication, and computation capabilities of end devices and edge servers to bring computing power closer to where data are produced. To improve the scalability of edge computing, next-generation communication systems are envisioned to provide energy transfer and harvesting for energy-constrained end devices, e.g., through 6G base stations (BSs) [4].

However, the wireless charging technology is relatively nascent with variable charging efficiencies [5]. For example, while magnetic resonant coupling has higher charging efficiency as compared to electromagnetic (EM) radiation power transfer, it is found in an earlier study that a 60 W power transfer over 2 meters can only be completed with 40% efficiency [6]. As such, apart from relying solely on wireless power charging enabled local computation, the resource-constrained IoT devices can also offload their computation tasks to the edge servers. Offloading computation-intensive tasks from the IoT devices to the edge servers helps to enhance the performance of the IoT devices while minimizing the device energy usage [7]. The IoT devices can leverage the resources of multiple edge servers to perform distributed computation tasks collaboratively. In order to mitigate the straggler effects in the distributed computing network

where the computation tasks are delayed by the slower edge servers, coding techniques [8] can be used to split a computation task into multiple subtasks, which are in turn allocated to the edge servers. In particular, the IoT devices can complete the computation tasks upon receiving the computed results from a subset, instead of all edge servers. Moreover, coding redundancy alleviates the need to wait for the tail-end straggling edge servers to complete their allocated tasks, hence reducing the latency of the computation tasks.

In this paper, we propose a wireless powered hybrid coded edge computing network in which IoT devices, i.e., users, can perform their computation tasks using one of the following approaches:

1) *Full local computation:* Each cell of users is supported by a base station that powers the users through wireless power charging, thereby enabling the users to locally complete the computation tasks. However, the charging efficiency and resultant computation shortfall are only known *ex-post*.
2) *Full coded offloading:* Each user utilizes coding techniques to allocate the computation subtasks to the dedicated or non-dedicated edge servers for computation, thereby completing their task while reducing the computation latency. Note that a non-dedicated edge server, e.g., Docker [9], is cheaper to deploy as it shares its computation resources with other applications [10]. However, the computation process is delayed if it is occupied by too many users simultaneously, and this knowledge of its computation capacity is only known ex-post. In contrast, a dedicated edge server can be used exclusively by a requester [11], e.g., the Azure private multi-access edge compute [11], but is more costly. If the non-dedicated edge servers are occupied, the subtasks have to be re-offloaded to the dedicated edge servers. This is known as *corrective edge offloading*.
3) *Hybrid approach:* Each user partially offloads to dedicated/non-dedicated edge servers while simultaneously performing local computation.

To minimize the network cost while accounting for the two sources of uncertainty (i.e., charging efficiency and edge computation capacity uncertainty), we utilize a stochastic resource optimization approach. We formulate the computation decision problem into a two-stage stochastic integer programming (SIP) model. In the first stage, the computation decision is made. In the second stage, the corrective edge offloading decision can be determined if there is a shortfall, which is known ex-post. As the SIP problem may become computationally intractable since it scales with the size of the network, we introduce the Benders' decomposition and sample average approximation (SAA) to solve the SIP problem efficiently. Moreover, we consider the special case of z-stage SIP in which corrective edge offloading actions can be implemented over multiple stages for less time-sensitive tasks.

The contributions of our paper are summarized as follows:

1) We present a novel wireless powered hybrid coded edge computing network that jointly leverages the use of base stations as wireless charging stations and edge servers for coded computation offloading. The network caters to energy-constrained IoT devices to perform computation-intensive tasks. Moreover, the coded edge offloading mitigates the stragglers effect of distributed computation.
2) We propose a stochastic resource optimization solution for efficient resource management amid stochastic uncertainties that can only be realized ex-post. Our solution is able to derive the lowest cost of the network while accounting for the two sources of uncertainty in terms of wireless power charging inefficiency and unsuccessful edge offloading.
3) We analyze and reduce the complexity of the two-stage SIP model using Benders' decomposition and sample-average approximation. As a result, the solution is computationally tractable even as the network scales. In consideration of less time-sensitive computation tasks, we also introduce the special case using z-stage that enables corrective edge offloading over multiple stages.

The remainder of the paper is organized as follows. Section II reviews the related work. Section III presents the system model. Sections IV and V discuss the two-stage and $z-$ stage SIP models respectively. Then, we evaluate the performance of the proposed framework in Section VI, followed by the conclusion in Section VII.

II. Related Work

Given the exponential increase in the amount of data generated, the energy-constrained IoT devices are no longer able to handle computation-intensive tasks. There are many studies that investigate effective ways to complete these computation tasks, which provide important insights into many data analytic applications. One of the approaches is to offload the computation-intensive tasks to the cloud, thereby boosting the performance of the IoT devices. For example, *Cuckoo* [12] and *COSMOS* [13] systems are proposed to bridge the demand of the individual mobile devices and the cloud resources such that the cost of providing the cloud resources is minimized while ensuring the computation speedup of the mobile applications. However, since the cloud may be located far away from the IoT devices, high latency and bad quality of experience are often the challenges for time-sensitive applications. As such, edge computing has been proposed where the cloud resources are extended to the edge of the networks, thereby resulting in a paradigm shift from cloud computing toward edge computing [14]. Several fundamental issues such as the utilization of resources [15], [16], [17], system architectures [18] and design of incentive mechanisms [19], [20] are actively studied in the literature in order to optimize the performance of the IoT devices in offloading their computation tasks. The authors in [21] have proposed a deep learning-based optimal offloading policy in order to minimize the computation and offloading overheads and gave information on the offloading tasks and network conditions. In order to ensure data integrity, a blockchain-enabled computation offloading scheme, which is known as *BeCome* is proposed [22].

Instead of offloading their computation tasks to a single edge node, which causes single-point-of-failure problems, IoT devices can leverage the resources of multiple edge nodes.

However, the main challenge of completing the distributed computation tasks is the straggler effects where the computation tasks are delayed significantly by the stragglers due to several factors such as contention of resources and hardware failure [23], [24]. In response to this, coding techniques can be used to mitigate the straggler effects by reducing the recovery threshold. In particular, the IoT devices can complete the computation tasks upon receiving the computed results from a subset of edge nodes, instead of all of them. Several coding schemes such as Polynomial codes [25], Entangled Polynomial codes [26], PolyDot codes [27], Short-dot codes [28], and Lagrange codes [29] are proposed to minimize the recovery threshold for distributed matrix multiplication problems. For gradient descent problems, fractional and cyclic repetition coding [30] are used. Instead of discarding the work done by the stragglers, several coding schemes exploit the work done by the stragglers through sequential processing [31] and multi-message communication [32]. This is especially useful when there exist non-persistent stragglers in the network. In order to improve the performance of the distributed computing networks, another research direction focuses on the minimization of computation load. The authors in [33] investigate the use of shift-and-addition encoding and zigzag decoding to avoid expensive multiplication and division operations in order to tackle the straggler problems more efficiently.

Another approach to complete the computation tasks is to gain the required energy through wireless charging. Wireless energy transfer will become a key feature of next-generation wireless networks given the versatility it brings towards empowering low energy devices for performing compute-intensive tasks. One such example is SoftBank's 5 G base stations[1] which are capable of energy transfer of 1 mW of power to users over high-frequency bands. Due to the mobility and flexibility of the Unmanned Aerial Vehicles (UAVs), they have been deployed as mobile base stations with wireless energy transfer capabilities as well [34].

However, it is noted that wireless energy transfer using EM radiation has poor efficiencies as a result of relative position changes and physical obstructions [5]. In fact, even the wireless energy transfer using magnetic resonant coupling has several drawbacks [35]. However, these inefficiencies are only realized ex-post. As such, it is instrumental for network cost minimization approaches to take into account the stochastic nature of such inefficiencies in deciding the computation strategies for each user.

III. System Model

We consider a wireless power charging enabled coded edge computing network (Fig. 1) with a coverage area that is divided into I cells, represented by a set $\mathcal{I} = \{1, \ldots, i, \ldots, I\}$. Each cell i contains W_i IoT devices, i.e., users, where the total number of users in each cell i may be different. Let d_{iw} denote the w^{th} user in cell i. The users aim to perform machine learning computation tasks using the data collected from their surroundings. Given the large amounts of data collected, the users may not have sufficient computation resources, e.g., energy, to complete the machine learning tasks individually.

In general, the hybrid network enables users to complete the computation tasks using the following approaches:

1) *Full local computation:* The I users are supported by H BSs, the set of which is denoted by $\mathcal{L} = \{1, \ldots, l, \ldots, L\}$ which can power the users through wireless power charging. The wireless power charging allows the users to have sufficient computation power to complete the machine learning tasks locally. While BSs are able to transfer their power to the users in a wireless manner, they may not always be reliable due to the variability in the efficiency of the magnetic coupling resonant. More specifically, the fluctuation of resonance frequency affects the efficiency of power transfer from the BSs to the users [36].
2) *Full coded offloading:* The I users are supported by K non-dedicated and X dedicated edge servers which are represented by sets $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$ and $\mathcal{X} = \{1, \ldots, x, \ldots, X\}$, respectively. The edge servers provide coded computation offloading support for the users. Specifically, the users use coding techniques, e.g., Polynomial codes [25], to allocate the distributed computation subtasks to the edge servers that have sufficient resources. The use of coding techniques minimizes the computation latency by reducing the recovery threshold, which is defined as the number of edge servers that need to return their computed results for the users to reconstruct the final result. However, the non-dedicated edge server may be occupied and congested, and thus the edge offloading fails. In this case, the corrective edge offloading of the computation task to dedicated edge servers has to be implemented.
3) *Hybrid approach:* The users may adopt a combined approach in some cases, e.g., adopt wireless charging for *partial* local computation while simultaneously offloading the other subtasks to the edge servers.

In the following subsections, we discuss the BS charging model and coded distributed computing model. Then, we present the stochastic resource optimization problem formulation and optimization strategy adopted to achieve the objective of cost minimization in the hybrid network.

A. BSs Charging Model

To provide wireless power charging, BS l can provide P_{il}^{ower} power to charge users in cell i and the power is expressed as follows [37]:

$$P_{il}^{ower} = \sum_{w \in W_i} (\| e_{iwl} \|_2)^2, \tag{1}$$

and:

$$e_{iwl} = \frac{\bar{\beta}_l}{\bar{d}_{w,i,l}} \left[\cos\left(\frac{2\pi}{\kappa_l} \bar{d}_{w,i,l} + \phi_l \right) - j \sin\left(\frac{2\pi}{\kappa_l} \bar{d}_{w,i,l} + \phi_l \right) \right], \tag{2}$$

where $\bar{\beta}_l = \sqrt{\frac{Z_0 G_h P_l^o}{4\pi}}$, Z_0 is a physical constant indicating the wave-impedance of a plane wave in free space, G_l is the gain of

1.https://asia.nikkei.com/Business/SoftBank2/SoftBank-plans-wireless-power-grid-to-charge-smartwatches-earbuds

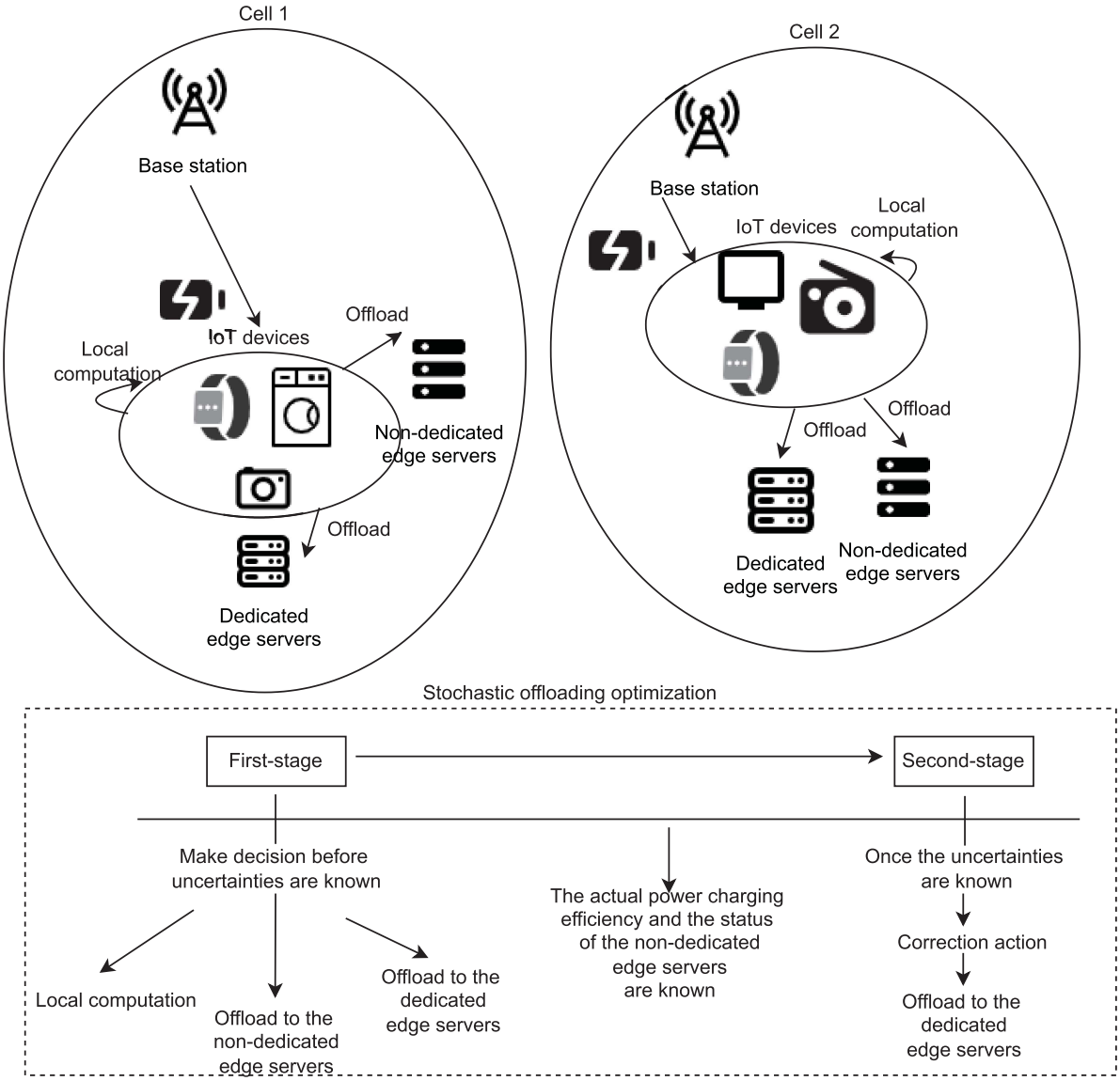


Fig. 1. The system model of multi-tier computing.

BS l, and P_l^o is the output power of BS l. κ_l is the wavelength of energy beam transmitted from BS l and $\bar{d}_{w,i,l}$ is the distance between BS l and d_{iw} user in cell i. (2) is a complex expression and the second term is the imaginary part. ϕ_l is the phase shift of the charging wave transmitted by BS l.

The power of the BSs is transferred to the users in the cell through the magnetic resonant coupling technique which allows high transmit efficiency over a relatively large distance. However, the performance of the BSs may be hindered by the uncertainty in the efficiency of the magnetic coupling resonant, which in turn affects the amount of energy that can be gained by the users. There are several factors that affect the efficiency of the magnetic coupling resonant such as the change in resonance frequency and the load value of the users [36]. In this paper, we denote the efficiency of the magnetic coupling resonant when BS l charges cell i by β_l^i. The values of β_l^i, $\forall i \in \mathcal{I}$ and $\forall l \in \mathcal{L}$, are between 0 and 1. For example, $\beta_l^i = 0.4$ indicates that 40% of the charging power of BS l can be transferred efficiently to user i.

The efficiency of the magnetic coupling resonant cannot be known precisely before the wireless power charging is completed. In a large-scale heterogeneous network, there may exist varying combinations of charging efficiencies across the different BSs. In this paper, we refer to each of these combinations as *scenarios*. Specifically, each scenario λ_y refers to a particular scenario indexed y, and is represented by a matrix of charging efficiencies $\beta_l^i(\lambda_y)$, $\forall i \in \mathcal{I}$ and $\forall l \in \mathcal{L}$. This matrix is expressed as follows:

$$\lambda_y = \begin{bmatrix} \beta_1^1(\lambda_y) & \beta_1^2(\lambda_y) & \cdots & \beta_1^I(\lambda_y) \\ \beta_2^1(\lambda_y) & \beta_2^2(\lambda_y) & \cdots & \beta_2^I(\lambda_y) \\ \vdots & \vdots & \ddots & \vdots \\ \beta_L^1(\lambda_y) & \beta_L^2(\lambda_y) & \cdots & \beta_L^I(\lambda_y) \end{bmatrix}. \quad (3)$$

The set of scenarios are represented by $\Lambda = \{\lambda_1, \ldots, \lambda_y, \ldots, \lambda_Y\}$ where Y is the total number of possible scenarios. The probability that scenario λ_y happens is denoted as $P(\lambda_y)$. For the rest of the paper, we use $\beta_l^i(\lambda_y)$ and β_l^i interchangeably where confusion does not arise from dropping the (λ_y) notation. Accordingly, the available power to be gained by user i from BS l, which is denoted as p_{il}^v, is expressed as follows:

$$p_{il}^v = \beta_l^i P_{il}^{ower}. \tag{4}$$

Besides local computation, we assume that the amount of power P_{il}^{ower} required by users in cell i also includes the power for data transmission.

B. Coded Distributed Computing Model

The users can also partially or completely offload their subtasks to the edge servers, where the resources of multiple edge servers are leveraged to complete the distributed computation tasks collaboratively. One of the main challenges of a distributed computing network is the straggler effects where the overall computation time is greatly increased by the straggling edge servers. Specifically, the distributed computation tasks are only completed after the slowest edge server completes its allocated computation task. In order to mitigate the straggler effects, coding techniques are used to split the computation tasks into smaller tasks, i.e., subtasks, and distribute them to the edge servers for computation. The objective of the coding schemes is to minimize the recovery threshold α_{iw}, which is defined as the number of edge servers that are required to return their computed results to the users in order to obtain the final result.

In many AI-based applications such as scientific computing, machine learning, and graph processing, matrix multiplication has been widely used. In this paper, we consider that each of the user d_{iw} adopts the Polynomial codes [25] to perform the distributed matrix multiplication computations,[2] i.e., $\mathbf{C}^{iw} = \mathbf{A}^{iw\top}\mathbf{B}^{iw}$ where $\mathbf{A}^{iw}$ and $\mathbf{B}^{iw}$ are input matrices of user d_{iw}, where $\mathbf{A}^{iw} \in \mathbb{F}_q^{s\times r}$ and $\mathbf{B}^{iw} \in \mathbb{F}_q^{s\times t}$ for integers s, r, and t and a sufficiently large finite field $\mathbb{F}_q$. Note that the Polynomial codes used in this paper can be extended by replacing Entangled Polynomial codes [26], PolyDot [27], and Lagrange codes [29], and it will change the recovery threshold. Entangled Polynomial codes is a generalized version of polynomial code. It allows column-wise partitioning of matrices with a parameter of $\hat{p}$, and it will achieve the recovery threshold of $\hat{p}mn + \hat{p} - 1$. When $\hat{p} = 1$, Entangled Polynomial codes equals Polynomial codes. PolyDot codes provides a trade-off between communication costs and recovery thresholds. A trade-off can be achieved by varying the value of s and t. The corresponding recovery threshold is $t^2(2s-1)$. Lagrange codes is a batch processing approach for coded distributed matrix multiplication. Sub-matrices can be grouped into $\mathcal{K}$ blocks, where $\mathcal{K} = \frac{\hat{n}}{g}$. $\hat{n}$ is the total number of workers and each worker stores g coded sub-matrices.To perform the coded distributed computation using Polynomial codes, the four important steps are as follows [38]:

1) *Distribution of computation tasks:* In this sub-section, we use k and x interchangeably where they are both referred to as the edge server. Given that the edge servers are able to store up to $\frac{1}{m_{iw}}$ and $\frac{1}{n_{iw}}$ fractions of matrices $\mathbf{A}^{iw}$ and $\mathbf{B}^{iw}$ respectively, user d_{iw} divides the input matrices into submatrices $\tilde{\mathbf{A}}_k^{iw} = f_k(\mathbf{A}^{iw})$ and $\tilde{\mathbf{B}}_k^{iw} = g_k(\mathbf{B}^{iw})$, where $\tilde{\mathbf{A}}_k^{iw} \in \mathbb{F}_q^{s\times \frac{r}{m_{iw}}}$ and $\tilde{\mathbf{B}}_k^{iw} \in \mathbb{F}_q^{s\times \frac{t}{n_{iw}}}$ respectively. Specifically, $\mathbf{f}$ and $\mathbf{g}$ represent the vectors of functions such that $\mathbf{f} = (f_1, \ldots, f_k, \ldots, f_K)$ and $\mathbf{g} = (g_1, \ldots, g_k, \ldots, g_K)$, respectively. Then, the user encodes and distributes the submatrices to the edge servers over the wireless channels for computations. Note that each edge server k may only receive one subtask, whereas the number of subtasks offloaded to the edge servers is based on the cost minimization optimization discussed in Section IV. Overall, the system model of the Polynomial codes consists of the followings [25]:
   - The user receives computation inputs, encodes them, and distributes them to the edge servers.
   - Edge servers perform pre-determined computations on their respective inputs in parallel.
   - The user receives outputs from successful worker nodes and decodes them to recover the final output.

   Note that data encoding and decoding actions exist in both local and offloading computation. Therefore, the energy consumption for data encoding and decoding is negligible as it exists in both.
2) *Computation by the edge servers:* Each edge server k is allocated submatrices $\tilde{\mathbf{A}}_k^{iw}$ and $\tilde{\mathbf{B}}_k^{iw}$ by the user. Based on the allocated submatrices, the edge servers perform the matrix multiplication, i.e., $\tilde{\mathbf{A}}_k^{iw\top}\tilde{\mathbf{B}}_k^{iw}$.
3) *Transmission of computed results:* Upon completion of the local computations, each edge server transmits its computed results, i.e., $\tilde{\mathbf{C}}_k^{iw} = \tilde{\mathbf{A}}_k^{iw\top}\tilde{\mathbf{B}}_k^{iw}$ to the user over the wireless communication channels.
4) *Reconstruction of final result:* By using the Polynomial codes, the user is able to decode and reconstruct the final result upon receiving α_{iw} computed results by using decoding functions. In other words, although the computation task is split and distributed to more than α_{iw} edge servers, the user does not need to wait for all the edge servers to return their computed results. The user only needs α_{iw} computed results where $\alpha_{iw} \leq K + X$. Note that although there is no constraint on the decoding functions to be used, a low-complexity decoding function such as the Reed-Solomon decoding algorithm [39] ensures the efficiency of the overall matrix multiplication computations.

Given that each edge server is able to store up to $\frac{1}{m_{iw}}$ and $\frac{1}{n_{iw}}$ fractions of matrices $\mathbf{A}^{iw}$ and $\mathbf{B}^{iw}$ respectively, the optimum recovery threshold that can be achieved by user d_{iw} using the Polynomial codes [25] is expressed as follows:

$$\alpha_{iw} = m_{iw}n_{iw}. \tag{5}$$

Compared with other coding techniques, such as MatDot codes [27], Polynomial codes have a lower computation cost per

2.Our system model can be easily extended to consider other types of distributed computation problems, e.g., gradient descent, Fourier transform and convolution as well as the multiplication that involves more than two matrices.

edge server and a higher communication cost from edge servers back to IoT devices, but they have a higher recovery threshold. However, we can carefully tune m_{iw} and n_{iw} from (5) to reduce the recovery threshold [27]. For example, if $m_{iw} = n_{iw} = 2$, the recovery threshold of Polynomial codes and MatDot codes are 4 and 3, respectively. For a square matrix of 36×36, the communication cost (symbols) of Polynomial codes and MatDot codes are 324 and 1296, respectively [25]. The computation cost (symbols) of Polynomial codes and MatDot codes are 11,664 and 23,328, respectively. Therefore, even though the recovery threshold of Polynomial codes is more than MatDot codes by 1, the overall number of symbols is lesser. The cost of deploying edge server k to complete the computation subtask is denoted as c_k. Hence, in order to complete its coded distributed computing task, user d_{iw} needs to pay a total amount of $\sum_{k \in \mathcal{L}_{iw}} c_k$, where $\mathcal{L}_{iw}$ represents the set of edge servers that successfully complete the computation subtasks allocated by user d_{iw}. For simplicity, we consider the cost of employing non-dedicated edge server k the same for all non-dedicated edge servers, i.e., $c_k = c, \forall k \in \mathcal{K}$. For example, the non-dedicated edge servers could belong to multiple service providers.

However, the desired response time is unable to be achieved when the non-dedicated edge server is used by multiple service providers at the same time [10]. Similarly, the cells do not know the information of whether other applications are using the non-dedicated edge servers *ex-ante*. The scenario of the non-dedicated edge server status is expressed as follows:

$$\eta_y = \{\gamma_1, \ldots, \gamma_K\}. \tag{6}$$

The offloaded subtasks have to be re-offloaded in this case to the dedicated edge server so that the computation process can be completed. The cost of employing the dedicated edge server x for corrective edge offloading is $\bar{c}$ and $\bar{c} > c$. The set of the current statuses of non-dedicated edge servers is denoted by $\eta_y \in \Omega$ and γ_k is a binary parameter. $\gamma_k = 0$, if the non-dedicated edge server k is fully occupied by another service provider and $\gamma_k = 1$ indicates otherwise. The system parameters are summarized in Table I.

TABLE I
SYSTEM MODEL PARAMETERS

Parameter	Description
$\mathbf{A}^{iw}$, $\mathbf{B}^{iw}$	Input matrices
α_{iw}	Recovery threshold of user
β_l^i	Efficiency of magnetic coupling resonant
$\underline{c}_{il}$	Cost of BS l for charging cell i
c_k	Cost of non-dedicated edge server k
$\bar{c}_x$	Cost of dedicated edge server x
c_i^p	Penalty cost
d_{iw}	w^{th} user in cell i
$\bar{d}_{w,i,l}$	Distance between BS l and user d_{iw}
G_l	Gain of BS l
I	Number of cells
K	Number of non-dedicated edge servers
κ_l	Wavelength of energy beam transmitted from BS l
l	Number of BSs
λ_y	Scenario
$\frac{1}{m_{iw}}$, $\frac{1}{n_{iw}}$	Fractions of input matrices
p_l^c	BS l maximum charging energy
p_{ij}^v	Available power to be gained
P_l^o	Output power of BS l
$P(\lambda_y)$	Probability that scenario y happens
ϕ_l	Phase shift of the charging wave transmitted by BS l
X	Number of dedicated edge servers
Z_0	Wave-impedance of a plane wave

IV. STOCHASTIC OFFLOADING OPTIMIZATION

In this section, we first consider the case where the wireless power charging efficiency and the current status of the non-dedicated edge servers are stochastic in nature, thereby necessitating the stochastic offloading optimization approach. Then, we introduce the deterministic integer programming approach (DIP), which is a benchmark (ideal but non-realistic) case when the power transfer efficiency and the status of the non-dedicated edge servers are precisely known ex-ante.

A. Two-Stage Stochastic Offloading Optimization

For easier representation, the uncertainty of charging efficiency from (3) and the uncertainty of the current status of the non-dedicated edge servers from (6) can be re-expressed as follows:

$$\lambda_y = \begin{bmatrix} F_1^1 & F_1^2 & \cdots & F_1^I \\ F_2^1 & F_2^2 & \cdots & F_2^I \\ \vdots & \vdots & \ddots & \vdots \\ F_L^1 & F_L^2 & \cdots & F_L^I \end{bmatrix}, \tag{7}$$

where $F_l^i = (\beta_l^i, \gamma_1, \ldots, \gamma_K)$. For example, $F_l^i = (\beta_l^i : 0.6, \gamma_1 : 0, \gamma_2 : 1, \gamma_3 : 1)$ means that the wireless power charging efficiency of BS l in cell i is 0.6. The non-dedicated edge server 1 is fully occupied while the non-dedicated edge servers 2 and 3 are available to be used. Similar to (3), the set of scenarios are represented by $\Lambda = \{\lambda_1, \ldots, \lambda_y, \ldots, \lambda_Y\}$.

To account for the stochastic nature of wireless power charging efficiency and the status of the non-dedicated edge servers, we formulate the cost minimization problem into a two-stage SIP model in this section [40].

The two stages are as follows:

- *First stage:* The network makes the ex-ante decision between i) full local computation, ii) full coded edge offloading, and iii) hybrid computation with partial local and partial offloading.

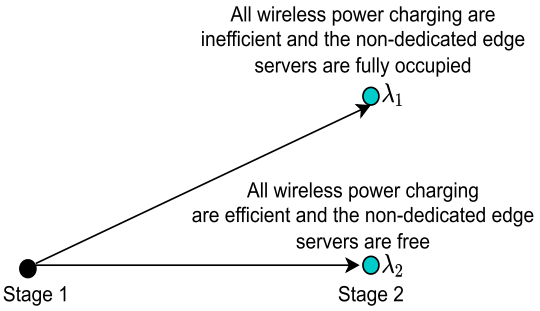


Fig. 2. The scenario tree of a two-stage SIP with the scenarios of λ_1 and λ_2.

- *Second stage:* The scenarios of wireless charging efficiency and non-dedicated edge servers availability are realized and observed. This means that each cell now has the full information on the charging efficiency of each BS and the status of the non-dedicated edge servers. If there is a shortfall, the cell makes the ex-post decision of corrective edge offloading to offload the remaining incomplete computation subtasks to the dedicated edge servers.

Using the SIP model, the network is able to alter its first stage ex-ante decision in response to the expected outcomes of the second. As an illustration, Fig. 2 shows a simplistic scenario tree with two scenarios in each stage. The scenarios in stage two are: i) experienced wireless power charging inefficiency and the non-dedicated edge servers are fully occupied by other applications, and ii) BSs have efficient wireless power charging and the non-dedicated edge servers are free. The scenario tree is a reduced form of an ensemble of scenarios or realizations of a process [41]. The tree clusters those realizations into a set of branches with specified probabilities of occurrence. Each scenario contains two nodes. In stage 1, all scenarios in the tree share the same node. In stage 2, the tree branches out and each branch belongs to only one scenario.

The two-stage SIP can be solved by generating a finite set of scenarios Λ with the likelihood of each scenario within the set occurring to be the corresponding probability $P(\lambda_y)$. The SIP problem can be solved efficiently with a reasonable accuracy provided that the following conditions are met [42]:

1) The number of linear constraints is fixed.
2) For every linear constraint and every possible realization of scenario, the second stage problem is feasible.
3) The number of scenarios is not too large.

In our problem formulation, condition 1 is satisfied. By the definition of feasibility given in [43], condition 2 is also satisfied since there is a corrective action to be selected in stage two to correct the eventual discrepancies in stage one. However, condition 3 may not be satisfied especially since the number of scenarios scales with the size of the network. As such, we utilize the Benders' decomposition and sample-average approximation in the subsequent subsections to solve the SIP in a computationally efficient manner. The decision variables are as follows:

- $\delta_{il} \in \{0,1\}$ indicates whether BS l is allocated to cell i. When $\delta_{il}=1$, BS l is allocated to cell i and $\delta_{il}=0$ indicates otherwise.
- $\bar{\delta}_{il} \in \{1,2,\dots,\}$ indicates the number of sub-tasks that are computed locally by the cell i when the power is provided by BS l. When $\bar{\delta}_{il}=1$, cell i computes 1 sub-task locally.
- $\mu_{ik} \in \{0,1\}$ indicates whether the sub-task is offloaded to the non-dedicated edge server k by cell i in stage one. Specifically, $\mu_{ik}=1$ if cell i offload 1 sub-task to the edge server k and $\mu_{ik}=0$ indicates otherwise.
- $\bar{\mu}_{ix} \in \{0,1\}$ indicates whether the sub-task is offloaded to the dedicated edge server x by cell i in stage one. Unlike edge server k, edge server x does not share its computation capability with other applications.
- $\bar{\mu}^{re}_{ix}(\lambda_y) \in \{0,1\}$ indicates whether the sub-task is re-offloaded to the dedicated edge server x by cell i in stage two under scenario λ_y.
- $\mu^p_i \in \{0,1\}$ indicates the implementation of the penalty cost c^p_i in stage 2. When $\mu^p_i=1$, cell i performs re-offloading action with an additional cost c^p_i and $\mu^p_i=0$ otherwise.

In total, there are four types of the costs as follows:

- c is the cost to employ a non-dedicated edge server k, i.e., the cost for edge offloading of a subtask in stage 1.
- $\underline{c}_{il}$ is the cost that BS h received from cell i, i.e., the cost for local computation, and it is expressed as follows:

$$\underline{c}_{il} = \alpha_0 p^v_{il}, \tag{8}$$

  where α_0 is the cost coefficient.
- $\bar{c}$ is the cost to employ dedicated edge server x and $\bar{c} > c$.
- c^p_i is the penalty cost for cell i that occurs in stage 2. It is the penalty cost for the time that is wasted when the wireless power charging is not efficient and a shortfall occurs that has to be corrected.

The objective function given in (9) and (10) is to minimize the total cost of the network. The expressions in (9) and (10) represent the first and second stage objectives, respectively. $\mathbb{E}\left[\mathcal{Q}(\bar{\mu}^{re}_{ix}(\lambda_y))\right]$ is the expectation over scenario $\lambda_y \in \Lambda$. The optimization variables, are the elements of the set $\Xi^{ts} = \{\delta_{il}, \bar{\delta}_{il}, \mu_{ik}, \bar{\mu}_{ix}, \mu^p_i(\lambda_y), \bar{\mu}^{re}_{ix}(\lambda_y)\}$. The SIP formulation can be expressed as follows:

$$\min_{\Xi^{ts}}: \sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\left(\delta_{il}+\bar{\delta}_{il}\underline{c}_{il}\right)+\sum_{i\in\mathcal{I}}\left(\sum_{k\in\mathcal{K}}\mu_{ik}c+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\bar{c}\right) +\mathbb{E}\left[\mathcal{Q}(\bar{\mu}^{re}_{ix}(\lambda_y))\right], \tag{9}$$

where

$$\mathcal{Q}(\bar{\mu}^{re}_{ix}(\lambda_y)) = \sum_{\lambda_y\in\Lambda} P(\lambda_y)\sum_{i\in\mathcal{I}}\left(\mu^p_i(\lambda_y)c^p_i+\sum_{x\in\mathcal{X}}\bar{\mu}^{re}_{ix}(\lambda_y)\bar{c}\right), \tag{10}$$

subject to

$$\sum_{l\in\mathcal{L}}\bar{\delta}_{il}\sum_{w\in W_i}\alpha_{iw}+\sum_{k\in\mathcal{K}}\mu_{ik}+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\geq\sum_{w\in W_i}\alpha_{iw}, \quad \forall i\in\mathcal{I}, \tag{11}$$

$$\sum_{l\in\mathcal{L}}\bar{\delta}_{il}\beta_l^i(\lambda_y)\sum_{w\in W_i}\alpha_{iw}+\sum_{k\in\mathcal{K}}\mu_{ik}\gamma_k(\lambda_y)+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{re}(\lambda_y)\geq\sum_{w\in W_i}\alpha_{iw},\quad\forall i\in\mathcal{I},\forall\lambda_y\in\Lambda,\tag{12}$$

$$\bar{\delta}_{il}\leq\bar{A}\delta_{il},\quad\forall i\in\mathcal{I},\forall l\in\mathcal{L},\tag{13}$$

$$\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{re}(\lambda_y)\leq\bar{A}\mu_i^p(\lambda_y),\quad\forall i\in\mathcal{I},\forall\lambda_y\in\Lambda,\tag{14}$$

$$\sum_{i\in\mathcal{I}}\delta_{il}\leq 1,\quad\forall l\in\mathcal{L},\tag{15}$$

$$\sum_{l\in\mathcal{L}}\delta_{il}\leq 1,\quad\forall i\in\mathcal{I},\tag{16}$$

$$\sum_{i\in\mathcal{I}}\mu_{ik}\leq 1,\quad\forall k\in\mathcal{K},\tag{17}$$

$$\sum_{i\in\mathcal{I}}\bar{\mu}_{ix}\leq 1,\quad\forall x\in\mathcal{X},\tag{18}$$

$$\sum_{i\in\mathcal{I}}\bar{\mu}_{ix}^{re}(\lambda_y)\leq 1,\quad\forall x\in\mathcal{X},\forall\lambda_y\in\Lambda,\tag{19}$$

$$\bar{\delta}_{il}p^s\sum_{w\in W_i}\alpha_{iw}\leq p_l^c,\quad\forall i\in\mathcal{I},\forall l\in\mathcal{L},\tag{20}$$

$$\delta_{il},\mu_{ik},\mu_{ix},\mu_i^p(\lambda_y),\bar{\mu}_{ix}^{re}(\lambda_y)\in\{0,1\},\quad\forall i\in\mathcal{I},\forall l\in\mathcal{L},\forall k\in\mathcal{K},\forall x\in\mathcal{X},\forall\lambda_y\in\Lambda,\tag{21}$$

$$\bar{\delta}_{il}\in\{0,\ldots,1\},\quad\forall i\in\mathcal{I},\forall l\in\mathcal{L}.\tag{22}$$

(11) and (12) ensure that the total number of sub-tasks that are computed should be greater than the total recovery threshold. If the non-dedicated edge servers are occupied, the sub-tasks are re-offloaded again in the next stage. (13) ensures that if cell i performs local computation, a BS should be allocated to cell i, where $\bar{A}$ is any large number. (14) ensures that the penalty cost is paid when the cell has to perform re-offloading. (15) and (16) ensure a one-to-one matching between the cell and the BS, i.e., a maximum of only one BS can be allocated to a cell, and a cell cannot choose another BS that has already been allocated. (17)–(19) ensures that the cell i cannot choose an occupied edger server when performing offloading and re-offloading. (20) ensures that the amount of energy that cell i uses should not exceed the maximum energy p_l^c that BS l provides and p^s is the energy used per sub-task. (21) indicates that the variables are binary variables. (22) indicates that $\bar{\delta}_{il}$ is a positive variable with a value between 0 and 1.

B. Benders' Decomposition

The problems formulated in (9) and (10) become complicated when the number of scenarios is large. The complexity of the problem depends on the number of variables, parameters, and constraints, which increases with the size of the network. SIP will potentially become computationally intractable if there are many scenarios [44]. Therefore, we propose the use of Benders' decomposition to decompose the original problem into multiple smaller problems [45]. Each of these can be solved iteratively to arrive at a final solution to the overall problem [45]. The benefit of this technique is that the decomposed problems are easier to solve than the original problem, so even if they have to be solved iteratively to obtain the solution, it will still be faster than trying to solve the original problem. However, (9) and (10) contain complicating variables that prevent the problem from being decomposed, i.e., first-stage variables [44]. Therefore, we can fix the complicating variables with a fixed value [45] so that the two-stage SIP can be decomposed into the independent optimization of master (23) and sub-problems (33).

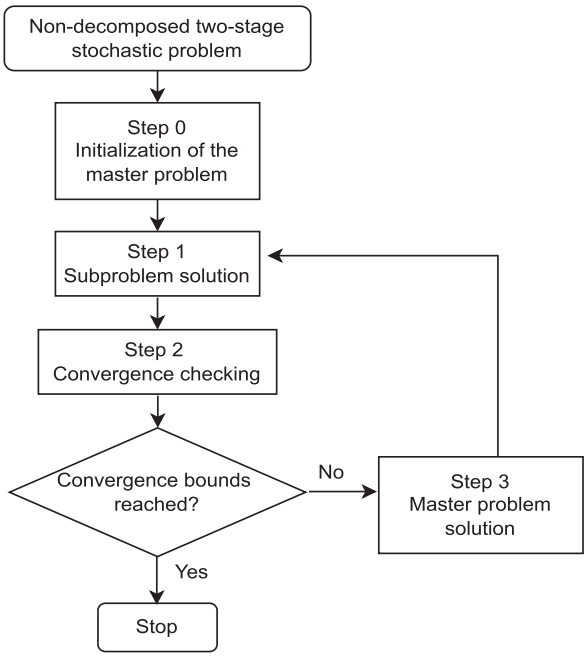


Fig. 3. Flow chart of Benders' decomposition algorithm.

The Benders' decomposition algorithm is performed iteratively to solve the decision variables and calculate the lower and upper bound. The algorithm terminates when the optimal solution converges, i.e., the difference between the lower bound and upper bounds are less than ϵ, i.e., a very small tolerance value. Fig. 3 shows the flowchart of Benders' decomposition algorithm. The algorithm mainly consists of four steps as follows.

Step 0: Initialization of the Master Problem. We initialize the parameters in the master problem once at the beginning of the algorithm. Let τ denote the iteration counter, v denote the iteration before τ and initially set $\tau=1$. The optimization variables, are the elements of the set $\Xi^{ms}=\{\delta_{il}^\tau,\bar{\delta}_{il}^\tau,\mu_{ik}^\tau,\bar{\mu}_{ix}^\tau\}$. The master problem is expressed as follows.

$$\min_{\Xi^{ms}}:\quad\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\left(\delta_{il}^\tau+\bar{\delta}_{il}^\tau\underline{c}_{il}\right)+\sum_{i\in\mathcal{I}}\left(\sum_{k\in\mathcal{K}}\mu_{ik}^\tau c+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^\tau\bar{c}\right)+\alpha_\tau,\tag{23}$$

subject to

$$\sum_{l\in\mathcal{L}}\bar{\delta}^{\tau}_{il}\sum_{w\in W_i}\alpha_{iw}+\sum_{k\in\mathcal{K}}\mu^{\tau}_{ik}+\sum_{x\in\mathcal{X}}\bar{\mu}^{\tau}_{ix}\geq\sum_{w\in W_i}\alpha_{iw},\quad\forall i\in\mathcal{I},\tag{24}$$

$$\bar{\delta}^{\tau}_{il}\leq\bar{A}\delta^{\tau}_{il},\qquad\forall i\in\mathcal{I},\forall l\in\mathcal{L},\tag{25}$$

$$\sum_{i\in\mathcal{I}}\delta^{\tau}_{il}\leq 1,\qquad\forall l\in\mathcal{L},\tag{26}$$

$$\sum_{l\in\mathcal{L}}\delta^{\tau}_{il}\leq 1,\qquad\forall i\in\mathcal{I},\tag{27}$$

$$\sum_{i\in\mathcal{I}}\mu^{\tau}_{ik}\leq 1,\qquad\forall k\in\mathcal{K},\tag{28}$$

$$\sum_{i\in\mathcal{I}}\mu^{\tau}_{ix}\leq 1,\qquad\forall x\in\mathcal{X},\tag{29}$$

$$\bar{\delta}^{\tau}_{il}p^{s}\sum_{w\in W_i}\alpha_{iw}\leq p^{c}_{l},\qquad\forall i\in\mathcal{I},\forall l\in\mathcal{L},\tag{30}$$

$$\alpha_{\tau}\geq\alpha^{(lb)},\tag{31}$$

$$\begin{aligned}\alpha_{\theta}\geq&\left[\sum_{\lambda_y\in\Lambda}P(\lambda_y)\sum_{i\in\mathcal{I}}\left(\mu^{(p,v)}_{i}(\lambda_y)c^{p}_{i}+\sum_{x\in\mathcal{X}}\bar{\mu}^{(re,v)}_{ix}(\lambda_y)\bar{c}\right)\right]\\&+\sum_{i\in\mathcal{I}}\sum_{\lambda_y\in\Lambda}\left(\sum_{l\in\mathcal{L}}\rho^{v}_{il}(\lambda_y)(\delta^{\tau}_{il}-\delta^{v}_{il})\right.\\&+\sum_{l\in\mathcal{L}}\bar{\rho}^{v}_{il}(\lambda_y)(\bar{\delta}^{\tau}_{il}-\bar{\delta}^{v}_{il})\\&+\sum_{k\in\mathcal{K}}\rho^{v}_{ik}(\lambda_y)(\mu^{\tau}_{ik}-\mu^{v}_{ik})\\&\left.+\sum_{x\in\mathcal{X}}\rho^{v}_{ix}(\lambda_y)(\mu^{\tau}_{ix}-\mu^{v}_{ix})\right).\end{aligned}\tag{32}$$

The objective function (23) and the constraints (24)–(30) are modified from (9), (11), (13), (15)–(18), and (20) by adding an iteration counter τ. α_τ is an auxiliary variable, representing the total objective function of the second-stage problem in iteration τ and it will be improved in subsequent iterations. $\alpha^{(lb)}$ is the lower bound of α_τ and it can be estimated from historical data of prior solutions [45]. (32) is the Benders' optimality cut and it does not exist when $\theta = 1$ as the variables and parameters are initialized with zero. Once the first-stage problem is solved, the algorithm proceeds to step 1. $\rho^{v}_{il}(\lambda_y)$, $\bar{\rho}^{v}_{il}(\lambda_y)$, $\rho^{v}_{ik}(\lambda_y)$, and $\rho^{v}_{ik=x}(\lambda_y)$ are the dual value of constraints (37)–(40), respectively.

Step 1: Sub-Problem Solution. Step 1 formulate and solve multiple sub-problems. For each scenario λ_y, the sub-problem is formulated as follows:

$$\min_{\mu^{(p,\tau)}_{i}(\lambda_y),\bar{\mu}^{(re,\tau)}_{ix}(\lambda_y)}:\quad P(\lambda_y)\sum_{i\in\mathcal{I}}\left(\mu^{(p,\tau)}_{i}(\lambda_y)c^{p}_{i}+\sum_{x\in\mathcal{X}}\bar{\mu}^{(re,\tau)}_{ix}(\lambda_y)\bar{c}\right),\tag{33}$$

subject to: (25),

$$\sum_{l\in\mathcal{L}}\bar{\delta}^{\tau}_{il}\beta^{i}_{l}(\lambda_y)\sum_{w\in W_i}\alpha_{iw}+\sum_{k\in\mathcal{K}}\mu^{\tau}_{ik}\gamma_{k}(\lambda_y)+\sum_{x\in\mathcal{X}}\bar{\mu}^{\tau}_{ix}+\sum_{x\in\mathcal{X}}\bar{\mu}^{(re,\tau)}_{ix}(\lambda_y)\geq\sum_{w\in W_i}\alpha_{iw},\quad\forall i\in\mathcal{I},\forall\lambda_y\in\Lambda,\tag{34}$$

$$\sum_{x\in\mathcal{X}}\bar{\mu}^{(re,\tau)}_{ix}(\lambda_y)\leq\bar{A}\mu^{(p,\tau)}_{i}(\lambda_y),\qquad\forall i\in\mathcal{I},\forall\lambda_y\in\Lambda,\tag{35}$$

$$\sum_{i\in\mathcal{I}}\bar{\mu}^{(re,\tau)}_{ix}(\lambda_y)\leq 1,\qquad\forall x\in\mathcal{X},\forall\lambda_y\in\Lambda,\tag{36}$$

$$\delta^{\tau}_{il}=\delta^{(\tau,fixed)}_{il},\qquad\forall i\in\mathcal{I},\forall l\in\mathcal{L},\tag{37}$$

$$\bar{\delta}^{\tau}_{il}=\bar{\delta}^{(\tau,fixed)}_{il},\qquad\forall i\in\mathcal{I},\forall l\in\mathcal{L},\tag{38}$$

$$\mu^{\tau}_{ik}=\mu^{\tau,fixed}_{ik},\qquad\forall i\in\mathcal{I},\forall k\in\mathcal{K},\tag{39}$$

$$\bar{\mu}^{\tau}_{ix}=\bar{\mu}^{\tau,fixed}_{ix},\qquad\forall i\in\mathcal{I},\forall x\in\mathcal{X}.\tag{40}$$

The objective function (33) and the constraints (34)–(36) are modified from (10), (12), (14), and (19) by adding an iteration counter τ. (37)–(40) is to fix the complicating variables with fixed variables.

Step 2: Convergence Checking. The master problem is the lower bound and the sub-problem is the upper bound of the solution. Both of the bounds are checked at the end of each iteration. Let $z^{(lb)}_{v}$ and $z^{(ub)}_{v}$ denotes the lower and upper bounds in iteration v from (23) and (33), respectively. The Benders' decomposition algorithm stops when $|z^{(lb)}_{v}-z^{(ub)}_{v}|<\epsilon$, which ϵ is a small tolerance value. Otherwise, the algorithm proceeds to the next iteration.

Step 3: First-Stage Problem Solution. In this step, the current iteration counter will be increased by 1. Then the master problem in (23) can be further relaxed by constraint (32). The solution to the master problem will also adjust α_τ, which is the on-demand cost. Once the master problem is solved, the algorithm proceeds to step 2 with the same iteration counter.

C. Sample-Average Approximation

The computation time of the master problem in (23) increases when more Benders' optimality cuts are added to the problem [46]. In this section, we use a different approach known as SAA to optimize stochastic problems when a large number of scenarios are used. We then compare the two methods in the performance evaluation section.

The SAA approach is about approximating the expected objective value utilizing sample scenarios Γ_N. $\Gamma_N = \{\lambda_1,\ldots,\lambda_n,\ldots,\lambda_N\}$ is the generated scenarios when the sample size is N. Instead of solving the problem using the true value, SAA approximates the true value by generating a sample of the stochastic parameters. This approach chooses a set of scenarios, e.g., N scenarios, where N is smaller than the total number of scenarios $|\Omega|$. The set of sample indices is represented by $\mathcal{N} = \{1,\ldots,n,\ldots,N\}$. This approach can obtain an optimal solution if N is large enough and thereby represents the scenarios

adequately, where N can be verified numerically. We revisit the numerical validation of N in Section VI.

The optimization variables, are the elements of the set $\Xi^{SAA} = \{\delta_{il}, \bar{\delta}_{il}, \mu_{ik}, \bar{\mu}_{ix}, \mu_i^p(\lambda_n), \bar{\mu}_{ix}^{re}(\lambda_n)\}$. The corresponding SAA objective function is expressed as follows:

$$\min_{\Xi^{SAA}}:$$

$$\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\left(\delta_{il}+\bar{\delta}_{il}\underline{c}_{il}\right)+\sum_{i\in\mathcal{I}}\left(\sum_{k\in\mathcal{K}}\mu_{ik}c+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\bar{c}\right)+\frac{1}{N}\sum_{n\in\mathcal{N}}\sum_{i\in\mathcal{I}}\left(\mu_i^p(\lambda_n)c_i^p+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{re}(\lambda_n)\bar{c}\right), \tag{41}$$

subject to: (11)–(22)

The SAA computation will first generate an N sample scenario and then the stochastic problem is solved by M times to produce a set of M possible solutions. The value of N and M are chosen in a way that the number of scenarios is large enough while yet maintaining the computability of the problems. When the same solution is found in these problems, we can choose the solution as the desired solution.

D. The Benchmark Case: Deterministic Offloading Optimization

In the ideal case, when the actual charging efficiency and the status of the non-dedicated edge servers are precisely known ex-ante, all cells can choose the correct action, whether to charge by the BSs or offload the tasks to the non-dedicated or dedicated edge server. The correction action is not required to perform and therefore, the total correction cost will be zero. The optimization variables, are the elements of the set $\Xi^{DIP} = \{\delta_{il}, \bar{\delta}_{il}, \mu_{ik}, \bar{\mu}_{ix}\}$. The optimization problem can be formulated by the DIP model [46], which is expressed as follows:

$$\min_{\Xi^{DIP}}:$$

$$\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\delta_{il}+\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\bar{\delta}_{il}\underline{c}_{il}+\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}}\mu_{ik}c+\sum_{i\in\mathcal{I}}\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\bar{c} \tag{42}$$

subject to: (13), (15)–(18), (20), (22),

$$\sum_{l\in\mathcal{L}}\bar{\delta}_{il}\bar{\beta}_l^i\sum_{w\in W_i}\alpha_{iw}+\sum_{k\in\mathcal{K}}\bar{\gamma}_k\mu_{ik}+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\geq\sum_{w\in W_i}\alpha_{iw},\qquad\forall i\in\mathcal{I}, \tag{43}$$

$$\delta_{il},\mu_{ik},\bar{\mu}_{ix},\in\{0,1\},\qquad\forall i\in\mathcal{I},\forall l\in\mathcal{L},\forall k\in\mathcal{K}, \tag{44}$$

where (43) ensures that the cell chooses a computation method, i.e., compute locally, offload to the non-dedicated edge servers, offload to the dedicated edge servers or perform both actions when the actual charging efficiency $\bar{\beta}_l^i$ and actual non-dedicated edge servers computation status $\bar{\gamma}_k$ are known. (44) indicates that the variables are binary.

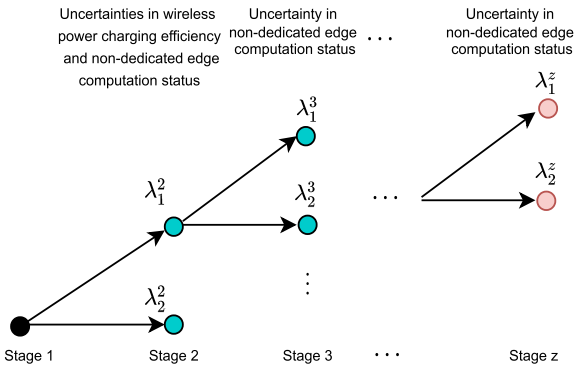


Fig. 4. The scenario tree of a z-stage SIP.

V. Case Study: Corrective Edge Offloading Decision Making Over Multiple Stages

The two-stage SIP is applicable when the cell has to complete its computation by the second stage. This means that the cell has to offload to a dedicated edge server at the second stage as it is unable to tolerate any uncertainty.

In contrast, we will study a special case whereby the cells are not urgent to implement their correction actions. For each stage after the first, the corrective action of re-offloading to the non-dedicated or dedicated edge servers can thus be performed. Specifically, the cheaper option of re-offloading to non-dedicated edge servers is made available due to the fact that the computation does not have to be completed by the second stage, and additional uncertainties can be tolerated.

We use a z-stage (multi-stage) SIP to formulate the decision of the cell in the z time steps. The uncertainty from (7) is added with a z symbol to indicate the scenario in each stage. The modified uncertainty in stage z is expressed as follows:

$$\lambda_y^z=\begin{bmatrix}F_1^{1,z} & F_1^{2,z} & \cdots & F_1^{I,z}\\ F_2^{1,z} & F_2^{2,z} & \cdots & F_2^{I,z}\\ \vdots & \vdots & \ddots & \vdots\\ F_L^{1,z} & F_L^{2,z} & \cdots & F_L^{I,z}\end{bmatrix}, \tag{45}$$

where $F_l^{i,z}=(\beta_l^{i,z},\gamma_1^z,\ldots,\gamma_K^z)$. The set of scenarios are represented by $\Lambda^z=\{\lambda_1^z,\ldots,\lambda_y^z,\ldots,\lambda_Y^z\}$. Since the uncertainty of charging efficiency is only used in stage 2, we can drop the $\beta_l^{i,z}$ from $F_l^{i,z}$ from stage 3 onward to reduce the number of parameters. As shown in Fig. 4, the z-stages are as follows:

- *First stage:* In the first stage, the network makes the ex-ante decision between i) full local computation, and ii) partial local computation. This decision is made before the scenarios are realized and observed. Note that the scenarios refer to the combinations as described in Section III-A.
- *Second stage:* In the second stage, the scenarios are realized and observed. In other words, each user now knows the shortfall in computation. To account for this shortfall, the user makes the ex-post decision to offload the remaining

incomplete computation subtasks to the dedicated edge servers or non-dedicated edge servers. Similarly, this decision is made before the second stage computation status of the non-dedicated edge servers can be realized.

- *Third stage:* In the third stage, the scenarios of the computation status of the non-dedicated edge servers from the second stage are realized and observed. If the computation is not complete (e.g., occupied non-dedicated edge servers in the second stage are used), the users are given an additional opportunity to offload to other or dedicated edge servers until the final stage z.

$\vdots$

- *z-stage:* In the z-stage, the scenarios of the computation status of the non-dedicated edge servers from $(z-1)$-stage are realized and observed. If the computation is not complete, the users are given a final opportunity to offload to other dedicated or non-dedicated edge servers, and the users are penalized by a large penalty cost $\tilde{c}$.

In our problem formulation, the decision variables are as follows:

- $\mu_{ik}^{(2)}(\lambda_y^2) \in \{0,1\}$ indicates whether the sub-task is offloaded to the non-dedicated edge server k by cell i in stage two and scenario λ_y^2. Specifically, $\mu_{ik}^{(2)}(\lambda_y^2) = 1$ if cell i offload 1 sub-task to the edge server k.
- $\bar{\mu}_{ix}^{(2)}(\lambda_y^2) \in \{0,1\}$ indicates whether the sub-task is offloaded to the dedicated edge server x by cell i in stage two and scenario λ_y^2. Specifically, $\bar{\mu}_{ix}^{(2)}(\lambda_y^2) = 1$ if cell i offload 1 sub-task to the edge server x.
- $\mu_i^{(2,p)}(\lambda_y^2) \in \{0,1\}$ indicates the implementation of the penalty cost c_i^p in stage 2. When $\mu_i^{(2,p)}(\lambda_y^2) = 1$, cell i performs re-offloading action in stage 2 with an additional cost of c_i^p and $\mu_i^{(2,p)}(\lambda_y^2) = 0$ means otherwise.

$\vdots$

- $\bar{\mu}_{ix}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z) \in \{0,1\}$ indicates whether the sub-task is re-offloaded to the dedicated edge server x by cell i in stage z.
- $\mu_{ik}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z) \in \{0,1\}$ indicates whether the sub-task is re-offloaded to the non-dedicated edge server k by cell i in stage z.
- $\mu_i^{(z,p)}(\lambda_y^2, \ldots, \lambda_y^z) \in \{0,1\}$ indicates the implementation of the penalty cost c_i^p in stage z.

The objective function given in (46)–(49) is to minimize the total cost of the network. The expressions in (46), (47), (48) and (49) represent the first, second, third, and z stage objectives, respectively. $\mathbb{E}\left[\mathcal{Q}(\mu_{ik}^{(2)}(\lambda_y^2))\right]$, $\mathbb{E}\left[\mathcal{Q}(\mu_{ik}^{(3)}(\lambda_y^2, \lambda_y^3))\right]$, ..., $\mathbb{E}\left[\mathcal{Q}(\mu_{ik}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z))\right]$ are the expectations over scenario $\lambda_y^2 \in \Lambda^2$, ..., $\lambda_y^z \in \Lambda^z$. $\tilde{c}$ is an additional large penalty cost that occurs in the final stage z when the network still has to perform a corrective action. The optimization variables, are the elements of the set $\Xi^z = \{\delta_{il}, \bar{\delta}_{il}, \mu_{ik}, \bar{\mu}_{ix}, \mu_{ik}^{(2)}(\lambda_y^2), \bar{\mu}_{ix}^{(2)}(\lambda_y^2), \ldots, \mu_i^{(z,p)}(\lambda_y^2, \ldots, \lambda_y^z)\}$. The SIP formulation can be expressed as follows:

$$\min_{\Xi^z}: \sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\left(\delta_{il} + \bar{\delta}_{il}\underline{c}_{il}\right) + \sum_{i\in\mathcal{I}}\left(\sum_{k\in\mathcal{K}}\mu_{ik}c + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\bar{c}\right) + \mathbb{E}\left[\mathcal{Q}(\mu_{ik}^{(2)}(\lambda_y^2))\right], \quad (46)$$

where

$$\mathcal{Q}(\mu_{ik}^{(2)}(\lambda_y^2)) = \sum_{\lambda_y^2\in\Lambda^2} P(\lambda_y^2)\sum_{i\in\mathcal{I}}\left(\sum_{k\in\mathcal{K}}\mu_{ik}^{(2)}(\lambda_y^2)c + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(2)}(\lambda_y^2)\bar{c} + \mu_i^{(2,p)}(\lambda_y^2)c_i^p + \mathbb{E}\left[\mathcal{Q}(\mu_{ik}^{(3)}(\lambda_y^2, \lambda_y^3))\right]\right), \quad (47)$$

$$\mathcal{Q}(\mu_{ik}^{(3)}(\lambda_y^2, \lambda_y^3)) = \sum_{\lambda_y^3\in\Lambda^3} P(\lambda_y^3)\sum_{i\in\mathcal{I}}\left(\sum_{k\in\mathcal{K}}c\mu_{ik}^{(3)}(\lambda_y^2, \lambda_y^3) + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(3)}(\lambda_y^2, \lambda_y^3)\bar{c} + \mu_i^{(3,p)}(\lambda_y^2, \lambda_y^3)c_i^p + \mathbb{E}\left[\mathcal{Q}(\mu_{ik}^{(4)}(\lambda_y^2, \lambda_y^3, \lambda_y^4))\right]\right), \quad \vdots \quad (48)$$

$$\mathcal{Q}(\mu_{ik}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z)) = \sum_{\lambda_y^z\in\Lambda^z} P(\lambda_y^z)\sum_{i\in\mathcal{I}}\left(\sum_{k\in\mathcal{K}} c\mu_{ik}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z) + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z)\bar{c} + \mu_i^{(z,p)}(\lambda_y^2, \ldots, \lambda_y^z)\tilde{c}\right), \quad (49)$$

subject to: (11), (13), (15)–(18), (20), (22),

$$\begin{aligned}
&\sum_{k\in\mathcal{K}}\mu_{ik}^{(2)}(\lambda_y^2)\gamma_k^2(\lambda_y^2)\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(2)}(\lambda_y^2) \geq \sum_{w\in W_i}\alpha_{iw} - \sum_{l\in\mathcal{L}}\bar{\delta}_{il}\beta_l^i(\lambda_y^2)\sum_{w\in W_i}\alpha_{iw} \\
&\qquad \forall i \in \mathcal{I}, \forall \lambda_y^2 \in \Lambda^2, \\
&\qquad \vdots \\
&\sum_{k\in\mathcal{K}}\left(\mu_{ik}^{(2)}(\lambda_y^2)\gamma_k^2(\lambda_y^2) + \mu_{ik}^{(3)}(\lambda_y^2, \lambda_y^3)\gamma_k^3(\lambda_y^2, \lambda_y^3) + \ldots + \mu_{ik}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z)\right) + \sum_{x\in\mathcal{X}}\left(\bar{\mu}_{ix}^{(2)}(\lambda_y^2) + \bar{\mu}_{ix}^{(3)}(\lambda_y^2, \lambda_y^3) + \ldots\right.
\end{aligned} \quad (50)$$

$$+\bar{\mu}_{ix}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\Big) \geq \sum_{w\in W_i}\alpha_{iw} - \sum_{l\in\mathcal{L}}\bar{\delta}_{il}\beta_l^i(\lambda_y^2)\sum_{w\in W_i}\alpha_{iw}$$
$$\forall i\in\mathcal{I},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z, \quad (51)$$

$$\sum_{k\in\mathcal{K}}\mu_{ik}^{(2)}(\lambda_y^2)+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(2)}(\lambda_y^2)\leq\bar{A}\mu_i^{(2,p)}(\lambda_y^2),$$
$$\forall i\in\mathcal{I},\forall\lambda_y^2\in\Lambda^2, \quad (52)$$

$$\vdots$$

$$\sum_{k\in\mathcal{K}}\mu_{ik}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)$$
$$\leq\bar{A}\mu_i^{(z,p)}(\lambda_y^2,\ldots,\lambda_y^z),$$
$$\forall i\in\mathcal{I},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z, \quad (53)$$

$$\sum_{i\in\mathcal{I}}\mu_{ik}^{(2)}(\lambda_y^2)\leq 1,\qquad \forall k\in\mathcal{K},\forall\lambda_y^2\in\Lambda^2, \quad (54)$$

$$\vdots$$

$$\sum_{i\in\mathcal{I}}\mu_{ik}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\leq 1,$$
$$\forall k\in\mathcal{K},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z, \quad (55)$$

$$\sum_{i\in\mathcal{I}}\bar{\mu}_{ix}^{(2)}(\lambda_y^2)\leq 1,\qquad \forall x\in\mathcal{X},\forall\lambda_y^2\in\Lambda^2, \quad (56)$$

$$\vdots$$

$$\sum_{i\in\mathcal{I}}\bar{\mu}_{ix}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\leq 1,$$
$$\forall x\in\mathcal{X},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z, \quad (57)$$

$$\delta_{il},\bar{\mu}_{ix}^{(2)}(\lambda_y^2),\ldots,\bar{\mu}_{ix}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\in\{0,1\},$$
$$\forall i\in\mathcal{I},\forall l\in\mathcal{L},\forall k\in\mathcal{K},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z. \quad (58)$$

(50) and (51) perform the same function as that of (11) and (12). (52) and (53) are the same as (14) to ensure that the penalty cost is paid when the cell requires to perform re-offloading. (54)–(57) are the same as (17)–(19). (58) indicates that the variables are binary variables. Similarly, when the number of scenarios are large, Benders' decomposition and SAA are applicable in the multi-stage SIP.

To solve the SIP, we assume that the probability distribution of all scenarios in set Λ^2,..., Λ^z are known [47], then, the complexity of the problem increases exponentially when the total number of scenarios across all the stages increases [47], [48]. The optimization algorithm is processed on the user terminal with their respective edge devices such as laptops and the algorithm is repeated for every computing tasks. By comparing the power usage for running optimization algorithm with the amount of power used in computation, computation consumes way more power. For example, a single V100 GPU can consume between 250 and 300 W [49], while the optimization algorithm consumes around 0.8 W (measured using open hardware monitor). Therefore, the energy consumption for optimization algorithm is negligible.

TABLE II
SYSTEM SIMULATION PARAMETER VALUES [37]

Parameter	Values
Phase shift of BS l, ϕ_l	$[0, 2\pi)$
Gain of BS l, G_l	12dB
Output power of BS l, P_l^o	10W
Cost of non-dedicated edge server c	\$1000
Cost of dedicated edge server $\bar{c}$	\$2000
Penalty cost c_i^p	\$500

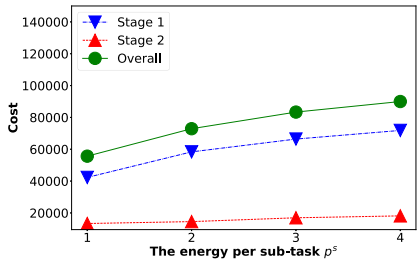


Fig. 5. The cost of the network by varying p^s.

VI. PERFORMANCE EVALUATION

The extensive experiments are conducted with two hundred edge servers, three cells, and three BSs. For the presented experiments, we implement the SIP model using GAMS script [50], [51]. The simulation parameters are presented in Table II. We first present the illustrative example of two scenarios. This enables us to understand the results when the parameters are varied. Then, we present the stochastic resource optimization when there are several (500) scenarios in more realistic network settings. Finally, we present the resource optimization over z-stages.

A. Illustrative Example of Two Scenarios

We vary α_{iw} randomly from 1 to 5 and set $p^s = 1$. Then, we consider two scenarios $|\Lambda| = 2$. The two scenarios are i) the wireless power charging efficiency from all the BSs is not efficient and the non-dedicated edge servers are fully occupied by other service providers λ_1, i.e., $\beta_l^i < 1$, and ii) all the wireless power charging efficiency are efficient and the non-dedicated edge servers are free λ_2, i.e., $\beta_l^i = 1$. The probabilities of each scenarios occurring are $P(\lambda_1) = 0.3$ and $P(\lambda_2) = 0.7$. We first use only scenarios λ_1 and λ_2 to perform the sensitivity test to evaluate the different parameters in a two-stage SIP.

1) Energy per Sub-Task p^s: We vary the energy per sub-task p^s and Fig. 5 shows the cost breakdown of the network. Among the three computation cost, $\underline{c}_{il}$, c and $\bar{c}$, the local computation is the cheapest. When $p^s = 1$, cells 1, 2, and 3 have sufficient energy to process most of the sub-tasks using three BSs. However, as the energy per sub-task increases, all the cells have to offload

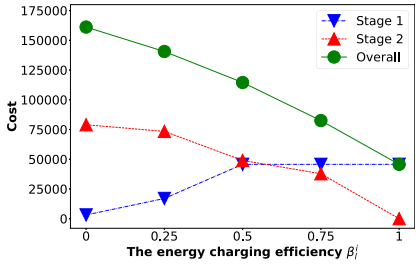


Fig. 6. The cost of the network by varying β_l^i.

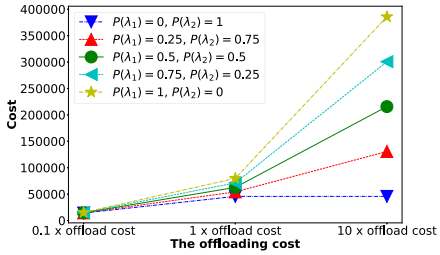


Fig. 7. The cost of the network by varying the offloading cost.

some sub-tasks to the non-dedicated edge servers. Therefore, the cost in stage 1 increases. The cost in stage 2 increases because when more sub-tasks are offloaded to the non-dedicated edge servers, more sub-tasks have to be re-offloaded whenever λ_1 occurs (i.e., the corrective edge offloading has to occur). Therefore, it is clear that all cells will use local computation where possible, when it is not limited by the energy constraint in (20).

2) Wireless Power Charging Efficiency β_l^i: We set $p^s = 0.5$, $P(\lambda_1) = 0.8$, $P(\lambda_2) = 0.2$, vary the wireless power charging efficiency β_l^i and keep the rest of the parameters same as Section VI-A1. The simulation result is shown in Fig. 6. When the wireless power charging efficiency is low ($\beta_l^i = 0$) all the cells except 3 will offload the sub-tasks to the edge servers, which means that cells 1 and 2 perform zero local computation. The cost is cheaper for cell 3 to perform the local computation first using BS 1 and then perform the re-offloading action to the dedicated edge servers when efficiency is low. As the efficiency increases, more cells are willing to perform local computation in stage 1 and re-offload in stage 2 if the efficiency is low. Therefore, the cost in stage 1 increases as efficiency increases, and the cost in stage 2 decreases as efficiency increases.

3) Probability Sensitivity: Similar to the setup in Section VI-A1, we observe the impact on the solution when varying both the wireless power charging efficiency probability $P(\lambda_1)$ and offloading cost. The result is shown in Fig. 7. We observe that the probability does not affect the decisions of all the cells when the offloading cost is low. All the cells will offload to the edge servers due to the cheap cost. As the offloading cost increases, the cost differences between each probability become larger. However, all the cells will still choose local computation even if the probability of power charging efficiency is low as the offloading cost is too high. The cells will perform local computation in the first stage and then offload the rest of the sub-tasks to the edge server when the efficiency is low.

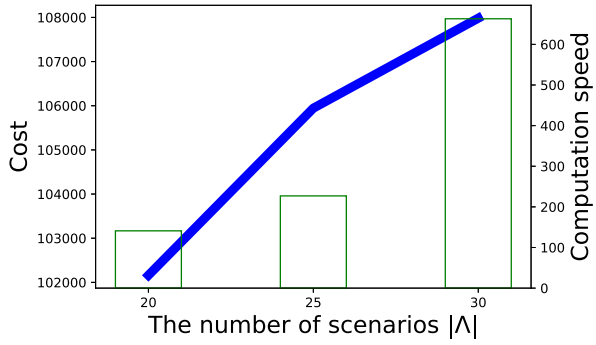


Fig. 8. The number of scenarios required.

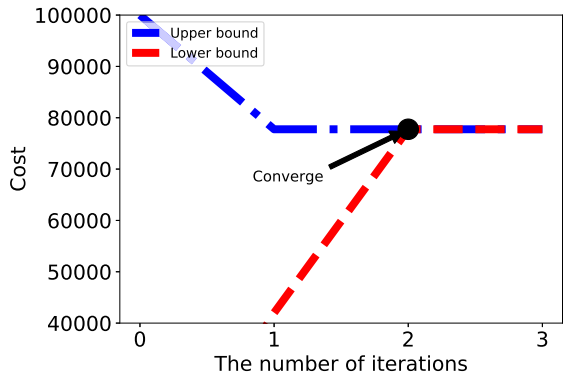


Fig. 9. The convergence of Benders' decomposition.

B. Stochastic Resource Optimization Under Several Scenarios

In the following, we consider when there are 500 scenarios.

1) Estimation of Number of Scenarios Required for SAA: We perform the simulation using the SAA approach to find the minimum number of scenarios required to best approximate the cost of the network. The simulations are performed $M = 2$ times. We first start the simulation by using 20 scenarios. The result is shown in Fig. 8. The bar chart that is shown below the line graph represents the computation speed in terms of seconds. The cost approximation depends on the sampled scenarios. When the number of scenarios are large enough, the approximated cost will be closer to the actual cost of the network [45]. However, when the number of scenarios increases, the complexity of the SAA solution increases. Therefore the computation time increases. In order to best approximate the network cost while balancing the computation cost/time, we choose to use 30 scenarios for the SAA approach unless otherwise stated for the rest of this section.

2) Convergence of Benders' Decomposition: Fig. 9 shows the bound convergence obtained by solving the Benders' decomposition algorithm. In initialization (iteration 0), the upper bound and the lower bounds are set at a very large positive and negative value, i.e., +10e10 and -10e-10, respectively. However, for better visualization, we limit the upper bound and the lower bound of

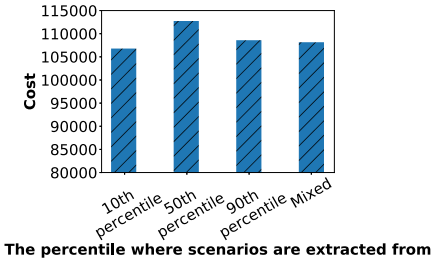


Fig. 10. Comparing the cost of SAA by using scenarios from different percentiles.

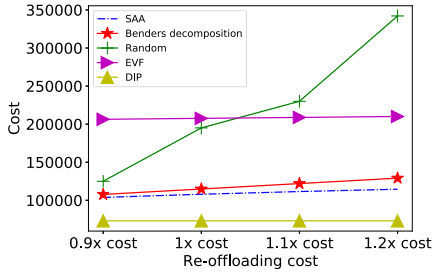


Fig. 11. Comparing between SIP scheme versus random scheme versus EVF scheme versus DIP scheme.

TABLE III
COMPUTATION SPEED OF SAA AND BENDERS' DECOMPOSITION IN TERM OF SECONDS

	SAA	Benders' decomposition (500 scenarios)
20 Scenarios	141s	173s
25 Scenarios	227s	
30 Scenarios	663s	

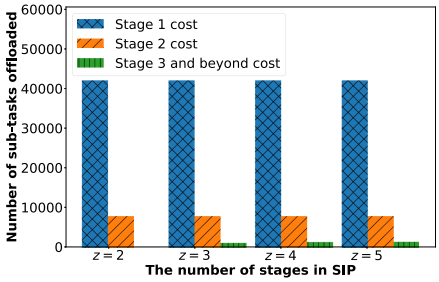


Fig. 12. The network cost when varying the number of stages z.

the figure to 100,000 and 40,000 respectively, and we extend the simulation by a few more iterations even if the algorithm converges. In each iteration, the upper bound and the lower bound are adjusted and they converge at the second iteration. The Benders' decomposition breaks down the two-stage SIP into the master and sub-problem. We observe that the algorithm can handle a large number of scenarios, i.e., 500, as the sub-problems can be solved efficiently due to their smaller number of variables and parallelization.

3) Extraction of Scenarios: We extract 30 scenario samples from three different percentiles of the distribution, head (10th), middle (50th) and tail (90th). The cost of the SAA is shown in Fig. 10. We can observe that the costs are different when the scenarios are extracted from different locations. This means that the cost from SAA is dependent on the sampled scenarios. Using the scenario generated from the middle portion of the distribution can lead to a high cost. However, the differences between them are not significant. The differences can be reduced when the scenarios are well sampled, as shown in the last (Mixed) column.

4) Computation Time Comparison Between SAA and Benders' Decomposition: We compare the computation time difference between SAA and Benders' decomposition. The result is shown in Table III. The computation time of the SAA tends to increase as the number of scenarios increases. Compared to the SAA, Benders' decomposition can maintain a much lower computation speed of 173 s while using a high number of scenarios (500).

5) Comparing Different Schemes: We compare the Benders' decomposition, SAA, DIP, expected-value formulation (EVF) [52] scheme, as well as the random scheme. The EVF uses the average value of the computation decision in stage 1 and solves an SAA with 30 scenarios. We use EVF as a fixed scheme and compare the results by varying the offloading price. For EVF, $\bar{\delta}_{il}$, μ_{ik} and $\bar{\mu}_{ix}$ are fixed using the average values of the computation decision from the historical records. The DIP is the ideal but unrealistic case in which we assume that the exact uncertainty of the wireless power charging efficiency and computation uncertainty for non-dedicated edge servers is exactly known. In a random scheme, the values of the decision variables are randomly generated. As shown in Fig. 11, both EVF and the random scheme derive the highest costs as they cannot adapt to the change in the corrective edge offloading cost. EVF is the highest when the costs are $0.9\times$ and $1\times$. The random scheme is the highest when the costs are $1.1\times$ and $1.2\times$. SAA which is an approximating solution, manage to achieve a relatively close cost as compared with the Benders' decomposition.

C. Resource Optimization Over z-Stages

Next, we will perform the simulation on z-stage SIP.

1) Number of Stages z: We first consider the case with two computation scenarios $|\Lambda^z| = 2$ for each of the SIP stages. The two scenarios are i) the wireless power charging from all the BSs is not efficient and some of the non-dedicated edge servers are occupied by other service providers λ_1^z, e.g., $\beta_l^{i\tilde{}} \neq \tilde{}1$, and ii) all the wireless power charging are efficient and the non-dedicated edge servers are free λ_2^z, e.g, $\beta_l^{i\tilde{}} = \tilde{}1$. The probabilities of uncertainty are $P(\lambda_1^z) = 0.3$ and $P(\lambda_2^z) = 0.7$. We vary the value of z from 2 to 5. Fig. 12 shows the cost breakdown, and the value of the decision variables are indicated in Table IV. The number of decision variables are large. Therefore, for illustration, Table IV does not contain any decision variables that have a value of zero. Then, we use the result from $z = 3$ to explain the findings. In stage 1, Cells 2 and 3 choose BS 1 and 3, respectively, to perform the local computation as the local computation is cheaper than the offloading. While cell 2 chooses to offload to the non-dedicated edge servers. When the efficiency of local computation and the uncertainty in computation status of the non-dedicated edge servers is low, cells 1, 2, and 3 will then

TABLE IV
DECISION VARIABLE VALUE

Variable	$z=2$	$\mathbf{z=3}$	$z=4$	$z=5$
δ_{11}	0	0	0	0
δ_{12}	0	0	0	0
δ_{13}	0	0	0	0
δ_{21}	1	**1**	1	1
δ_{22}	0	0	0	0
δ_{23}	0	0	0	0
δ_{31}	0	0	0	0
δ_{32}	0	0	0	0
δ_{33}	1	**1**	1	1
$\bar{\delta}_{11}$	0	0	0	0
$\bar{\delta}_{12}$	0	0	0	0
$\bar{\delta}_{13}$	0	0	0	0
$\bar{\delta}_{21}$	1	**1**	1	1
$\bar{\delta}_{22}$	0	0	0	0
$\bar{\delta}_{23}$	0	0	0	0
$\bar{\delta}_{31}$	0	0	0	0
$\bar{\delta}_{32}$	0	0	0	0
$\bar{\delta}_{33}$	1	**1**	1	1
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(2)}_{1k}(\lambda^2_1)$	13	**13**	13	13
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(2)}_{2k}(\lambda^2_1)$	6	**6**	6	6
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(2)}_{3k}(\lambda^2_1)$	6	**6**	6	6
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(3)}_{1k}(\lambda^2_1,\lambda^3_1)$	-	**6**	6	6
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(3)}_{2k}(\lambda^2_1,\lambda^3_1)$	-	**3**	3	3
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(3)}_{3k}(\lambda^2_1,\lambda^3_1)$	-	**3**	3	3
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{1k}(\lambda^2_1,\lambda^3_1,\lambda^4_1)$	-	-	3	6
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{2k}(\lambda^2_1,\lambda^3_1,\lambda^4_1)$	-	-	2	3
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{3k}(\lambda^2_1,\lambda^3_1,\lambda^4_1)$	-	-	2	3
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{1k}(\lambda^2_1,\ldots,\lambda^5_1)$	-	-	-	2
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{2k}(\lambda^2_1,\ldots,\lambda^5_1)$	-	-	-	1
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{3k}(\lambda^2_1,\ldots,\lambda^5_1)$	-	-	-	1

re-offload the remaining sub-tasks to the non-dedicated edge servers. Since $P(\lambda^z_1)$ is low at each SIP stage, it is cheaper for cells to re-offload the sub-tasks to non-dedicated edge servers than to dedicated edge servers. The cells can perform the correction action only when it is needed by offloading the sub-tasks to the non-dedicated edge servers.

2) Probability Sensitivity in z-Stage: We observe the impact on the network by varying both the probabilities $P(\lambda^z_1)$ and $P(\lambda^z_2)$ and the number of stages z. The cost of the network is shown in Fig. 13. If there is no uncertainty $P(\lambda^z_1)=0$, when z increases, the decision made by the cells remains the same. When $P(\lambda^z_1)$ increases, it means that the chances that the BS is not efficient and some of the non-dedicated edge servers are occupied by other service providers is high. Therefore, the cells will compute all the tasks in the earlier stage, i.e., stage one to avoid the high re-offloading cost.

3) Penalty Sensitivity: We consider a setup similar to Section VI-C1. We observe the cost of the network by varying the penalty cost of the re-offloading. The cost of the network is shown in Fig. 14. When the penalty cost is $1\times$ or $10\times$, all the cells will still perform re-offloading as the correction actions. However, all the cells change their decision when the penalty cost is $100\times$ as the correction action that happens at all the stages is higher than using the dedicated edge server in stage one.

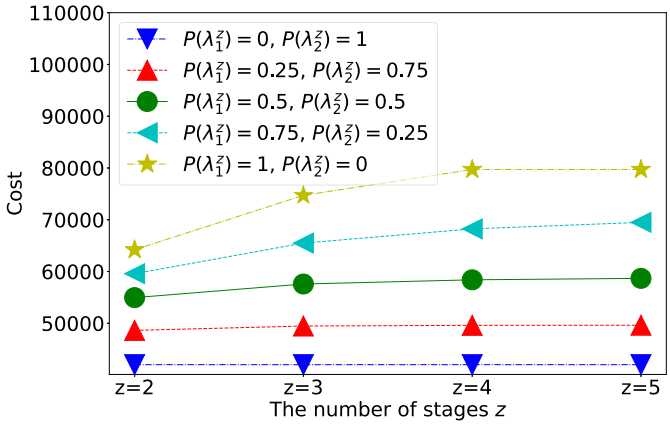


Fig. 13. The network cost when varying the probability of uncertainty in each stage.

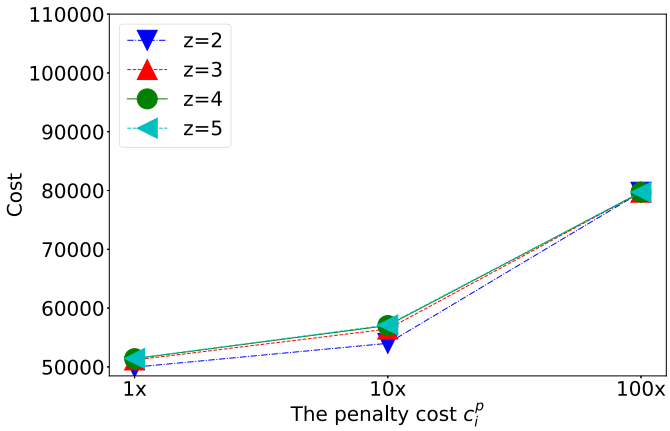


Fig. 14. The network cost when varying the penalty cost c^p_i.

VII. CONCLUSION

In this paper, we have introduced a stochastic resource optimization framework for the wireless powered hybrid coded edge computing network. Given the uncertainties of wireless charging efficiency and edge server availability, we have proposed a stochastic integer programming approach for resource optimization. As a result, we can obtain an optimal cost-minimizing computation strategy despite the fact that the uncertainties are only observed ex-post of making the computation decisions. For our future work, we will further incorporate other sources of uncertainties, such as malicious edge servers, into the problem formulation. This will further introduce additional properties of secure coded offloading into the problem formulation.

REFERENCES

[1] S. S. Mishra and A. Rasool, "IoT health care monitoring and tracking: A survey," in *Proc. 3rd Int. Conf. Trends Electron. Inform.*, Tirunelveli, India, 2019, pp. 1052–1057.
[2] C. Yang, T. Peng, S. Lan, W. Shen, and L. Wang, "Towards IoT-enabled dynamic service optimal selection in multiple manufacturing clouds," *J. Manuf. Syst.*, vol. 56, pp. 213–226, 2020.
[3] P. Ferrer-Cid, J. M. Barcelo-Ordinas, and J. Garcia-Vidal, "Graph learning techniques using structured data for IoT air pollution monitoring platforms," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13652–13663, Sep. 2021.

[4] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May/Jun. 2020.

[5] L. Xie et al., "A mobile platform for wireless charging and data collection in sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 8, pp. 1521–1533, Aug. 2015.

[6] A. Kurs, A. Karalis, R. Moffatt, J. D. Joannopoulos, P. Fisher, and M. Soljacic, "Wireless power transfer via strongly coupled magnetic resonances," *Science*, vol. 317, no. 5834, pp. 83–86, 2007.

[7] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

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

[9] Docker, "Docker pricing and subscriptions." Accessed: Apr. 14, 2022. [Online]. Available: https://www.docker.com/pricing/?utm_source=docker&utm_medium=webreferral&utm_campaign=traffictopricing

[10] M. Desertot, C. Escoffier, P. Lalanda, and D. Donsez, "Autonomic management of edge servers," in *Proc. Int. Workshop Self-Organizing Syst.*, Springer, 2006, pp. 216–229.

[11] M. Azure, "Azure private multi-access edge compute (MEC)." Accessed: Apr. 14, 2022. [Online]. Available: https://azure.microsoft.com/en-us/solutions/private-multi-access-edge-compute-mec/#overview

[12] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Proc. Int. Conf. Mobile Comput. Appl. Serv.*, M. Gris and G. Yang, Eds., Berlin, Heidelberg, Springer, 2012, pp. 59–79.

[13] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "COSMOS: Computation offloading as a service for mobile devices," in *Proc. 15th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Philadelphia, Pennsylvania, USA, 2014, pp. 287–296.

[14] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[15] N. Nouri, A. Entezari, J. Abouei, M. Jaseemuddin, and A. Anpalagan, "Dynamic power–Latency tradeoff for mobile edge computation offloading in NOMA-based networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2763–2776, Apr. 2020.

[16] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.

[17] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19324–19337, 2018.

[18] X. Wei et al., "MVR: An architecture for computation offloading in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput.*, Honolulu, Hawaii, USA, 2017, pp. 232–235.

[19] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach," *Comput. Netw.*, vol. 129, pp. 399–409, 2017.

[20] M. LiWang, S. Dai, Z. Gao, X. Du, M. Guizani, and H. Dai, "A computation offloading incentive mechanism with delay and cost constraints under 5G satellite-ground IoV architecture," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 124–132, Aug. 2019.

[21] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: A deep learning approach," in *Proc. IEEE 28th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, Montreal, Quebec, Canada, 2017, pp. 1–6.

[22] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, and W. Dou, "BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing," *IEEE Trans. Ind. Inform.*, vol. 16, no. 6, pp. 4187–4195, Jun. 2020.

[23] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[24] G. Ananthanarayanan et al., "Reining in the outliers in map-reduce clusters using Mantri," in *Proc. 9th USENIX Symp. Operating Syst. Des. Implementation*, Vancouver, British Columbia, Canada, 2010, pp. 265–278.

[25] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. Adv. Neural Inf. Process. Syst.*, I. U. V. Guyon, S. Luxburg, H. Bengio, R. Wallach, S. Fergus Vishwanathan, and R. Garnett, Eds., Long Beach, California, USA, Curran Associates, Inc., 2017, pp. 4403–4413.

[26] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1920–1933, Mar. 2020.

[27] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 278–301, Jan. 2020.

[28] S. Dutta, V. Cadambe, and P. Grover, ""Short-dot": Computing large linear transforms distributedly using coded short dot products," *IEEE Trans. Inf. Theory*, vol. 65, no. 10, pp. 6171–6193, Oct. 2019.

[29] 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. 22nd Int. Conf. Artif. Intell. Statist.*, PMLR, 2019, pp. 1215–1225.

[30] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. 34th Int. Conf. Mach. Learn.*, D. Precup and Y. W. Teh, Eds., International Convention Centre, Sydney, Australia, PMLR, 2017, pp. 3368–3376.

[31] N. Ferdinand and S. C. Draper, "Hierarchical coded computation," in *Proc. IEEE Int. Symp. Inf. Theory*, Vail, Colorado, USA, 2018, pp. 1620–1624.

[32] E. Ozfatura, D. Gündüz, and S. Ulukus, "Speeding up distributed gradient descent by utilizing non-persistent stragglers," in *Proc. IEEE Int. Symp. Inf. Theory*, Paris, France, 2019, pp. 2729–2733.

[33] M. Dai, Z. Zheng, S. Zhang, H. Wang, and X. Lin, "SAZD: A low computational load coded distributed computing framework for IoT systems," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3640–3649, Apr. 2020.

[34] H. Yan, Y. Chen, and S.-H. Yang, "UAV-Enabled wireless power transfer with base station charging and UAV power consumption," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12883–12896, Nov. 2020.

[35] S. D. Barman, A. W. Reza, N. Kumar, M. E. Karim, and A. B. Munir, "Wireless powering by magnetic resonant coupling: Recent trends in wireless power transfer system and its applications," *Renewable Sustain. Energy Rev.*, vol. 51, pp. 1525–1552, 2015.

[36] G. Zhang, C. Li, H. Zhang, and X. Jiang, "Parameters optimization for magnetic resonance coupling wireless power transmission," *Sci. World J.*, vol. 2014, pp. 1–8, 2014.

[37] Q. Sha, X. Liu, N. Ansari, and Y. Jia, "Efficient multiple charging base stations assignment for far-field wireless-charging in green IoT," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 1–6.

[38] J. S. Ng et al., "Collaborative coded computation offloading: An all-pay auction approach," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[39] F. Didier, "Efficient erasure decoding of reed-solomon codes," 2009, *arXiv:0901.1886*.

[40] S. Sawadsitang, R. Kaewpuang, S. Jiang, D. Niyato, and P. Wang, "Optimal stochastic delivery planning in full-truckload and less-than-truckload delivery," in *Proc. IEEE 85th Veh. Technol. Conf.*, 2017, pp. 1–5.

[41] L. S. Inc., "LINGO 17 online users manual." Accessed: Apr. 14, 2022. [Online]. Available: https://www.lindo.com/doc/online_help/lingo17_0/index.html?scenario_tree.htm

[42] A. Shapiro and A. Nemirovski, "On complexity of stochastic programming problems," in *Continuous Optimization*, Berlin, Germany: Springer, 2005, pp. 111–146.

[43] R. M. Van Slyke and R. Wets, "L-shaped linear programs with applications to optimal control and stochastic programming," *SIAM J. Appl. Math.*, vol. 17, no. 4, pp. 638–663, 1969.

[44] A. Schwele, J. Kazempour, and P. Pinson, "Do unit commitment constraints affect generation expansion planning? A scalable stochastic model," *Energy Syst.*, vol. 11, no. 2, pp. 247–282, 2020.

[45] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 164–177, Second Quarter 2012.

[46] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Proc. IEEE Asia-Pacific Serv. Comput. Conf.*, 2009, pp. 103–110.

[47] M. Dyer and L. Stougie, "Computational complexity of stochastic programming problems," *Math. Program.*, vol. 106, pp. 423–432, Dec. 2006.

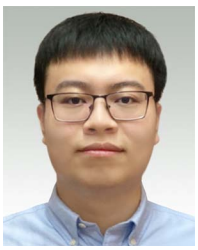
[48] S. Rajendran and A. R. Ravindran, "Platelet ordering policies at hospitals using stochastic integer programming model and heuristic approaches to reduce wastage," *Comput. Ind. Eng.*, vol. 110, pp. 151–164, 2017.

[49] M. Labbe, "Energy consumption of AI poses environmental problems." Accessed: Apr. 14, 2022. [Online]. Available: https://www.techtarget.com/searchenterpriseai/feature/Energy-consumption-of-AI-poses-environmental-problems

[50] D. Chattopadhyay, "Application of general algebraic modeling system to power system optimization," *IEEE Trans. Power Syst.*, vol. 14, no. 1, pp. 15–22, Feb. 1999.
[51] G. D. Corporation, "General algebraic modeling system (GAMS) release 24.2.1." Accessed: Apr. 14, 2022. [Online]. Available: http://www.gams.com/
[52] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Proc. IEEE Asia-Pacific Serv. Comput. Conf.*, 2009, pp. 103–110.

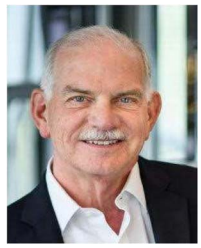
Wei Chong Ng received the BEng degree in electrical and electronic engineering (Highest Distinction) from Nanyang Technological University, Singapore, in 2020. He is currently working toward the PhD degree with Alibaba Group and Alibaba-NTU Joint Research Institute, Nanyang Technological University, Singapore. His research interests include the Metaverse, stochastic integer programming, and edge computing.

Wei Yang Bryan Lim received the PhD degree from Nanyang Technological University (NTU), Singapore, in 2022 under the Alibaba PhD Talent Programme. He is currently Wallenberg-NTU Presidential postdoctoral fellow.



Zehui Xiong received the PhD degree from Nanyang Technological University (NTU), Singapore. He is currently an assistant professor with the Singapore University of Technology and Design, and also an honorary adjunct senior research scientist with Alibaba-NTU Singapore Joint Research Institute, Singapore. He was the visiting scholar with Princeton University and University of Waterloo. His research interests include wireless communications, Internet of Things, blockchain, edge intelligence, and Metaverse. He has published more than 200 research papers in leading journals and flagship conferences and many of them are ESI Highly Cited Papers. He has won more than 10 Best Paper Awards in international conferences and is listed in the World's Top 2% scientists identified by Stanford University. He is now serving as the editor or guest editor for many leading journals including *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Vehicular Technology*, *IEEE Internet of Things Journal*. He is the recipient of IEEE Early Career Researcher Award for Excellence in Scalable Computing, IEEE Technical Committee on Blockchain and Distributed Ledger Technologies Early Career Award, IEEE Internet Technical Committee Early Achievement Award, IEEE TCI Rising Star Award, IEEE TCCLD Rising Star Award, IEEE Best Land Transportation Paper Award, IEEE CSIM Technical Committee Best Journal Paper Award, IEEE SPCC Technical Committee Best Paper Award, IEEE VTS Singapore Best Paper Award. He is now serving as the associate director of Future Communications R&D Programme.

Dusit Niyato (Fellow, IEEE) received the BEng degree from the King Mongkuts Institute of Technology Ladkrabang (KMITL), Thailand, in 1999 and the PhD degree in electrical and computer engineering from the University of Manitoba, Canada, in 2008. He is currently a professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests are in the areas of Internet of Things (IoT), machine learning, and incentive mechanism design.



H. Vincent Poor (Life Fellow, IEEE) received the PhD degree in EECS from Princeton University, in 1977. From 1977 until 1990, he was on the faculty of the University of Illinois at Urbana-Champaign. Since 1990 he has been on the faculty with Princeton, where he is currently the Michael Henry Strater University professor. During 2006 to 2016, he served as the dean of Princeton's School of Engineering and Applied Science. He has also held visiting appointments at several other universities, including most recently at Berkeley and Cambridge. His research interests are in the areas of information theory, machine learning and network science, and their applications in wireless networks, energy systems and related fields. Among his publications in these areas is the recent book Machine Learning and Wireless Communications. (Cambridge University Press, 2022). He is a member of the National Academy of Engineering and the National Academy of Sciences and is a foreign member of the Chinese Academy of Sciences, the Royal Society, and other national and international academies. He received the IEEE Alexander Graham Bell Medal, in 2017.

Xuemin Sherman Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, New Jersey, in 1990. He is a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, a Chinese Academy of Engineering Foreign member, and a distinguished lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT), in 2021, the R.A. Fessenden Award, in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario), in 2019, James Evans Avant Garde Award, in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award, in 2015 and Education Award, in 2017 from the IEEE Communications Society (ComSoc), and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award, in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA), in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee chair/co-chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, IEEE Globecom'07, and the chair for the IEEE ComSoc Technical Committee on Wireless Communications. He is the president of the IEEE ComSoc. He was the vice president for Technical & Educational Activities, vice president for Publications, Member-at-Large on the Board of Governors, chair of the distinguished lecturer Selection Committee, and member of IEEE Fellow Selection Committee of the ComSoc. He served as the editor-in-chief of the *IEEE IoT Journal*, *IEEE Network*, and *IET Communications*.

Chunyan Miao received the BS degree from Shandong University, Jinan, China, in 1988, and the MS and PhD degrees from Nanyang Technological University, Singapore, in 1998 and 2003, respectively. She is currently a professor with the School of Computer Science and Engineering, Nanyang Technological University (NTU), and the director of the Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY). Her research focus on infusing intelligent agents into interactive new media (virtual, mixed, mobile, and pervasive media) to create novel experiences and dimensions in game design, interactive narrative, and other real world agent systems.