SGCO: Stabilized Green Crosshaul Orchestration for Dense IoT Offloading Services

Nhu-Ngoc Dao, Duc-Nghia Vu, Woongsoo Na, Joongheon Kim, Sungrae Cho

***Abstract*—The next generation mobile network anticipates integrated heterogeneous fronthaul and backhaul technologies referred to as a unified crosshaul architecture. The crosshaul enables a flexible and cost-efficient infrastructure for handling mobile data tsunami from dense Internet of things (IoT). However, stabilization, energy efficiency, and latency have not been jointly considered in the optimization of crosshaul performance. To overcome these issues, we propose an orchestration scheme referred to as the stabilized green crosshaul orchestration (SGCO). SGCO utilizes a Lyapunov-theory-based drift-plus-penalty policy to determine the optimal amount of offloaded data that should be processed either at the eastbound or westbound computing platforms to minimize energy consumption. To achieve system stability, the cache buffer is considered as the main constraint in developing the optimization process. Moreover, the amount of offloaded data transmitted via crosshaul links is selected by adopting the binary min-knapsack problem. Accordingly, a lightweight heuristic algorithm is proposed. As the cache buffer is stabilized and the computations are controlled, the SGCO ensures adjustable computing latency threshold for various IoT services. The performance analysis shows that the proposed SGCO scheme exposes effective energy consumption compared to other existing schemes while maintaining system stability considering latency.**

***Index Terms*—crosshaul computing, energy efficiency, cache stability, latency awareness, dense IoT.**

I. Introduction

The exponential growth of big data generated from dense Internet of things (IoT) systems entails a variety of novel technologies in the fifth generation (5G) networks. Among these, network softwarization and cloudization are considered as two prime contributors, which support scalable network virtualization and computation offloading services for massive IoT devices, respectively [1]. Based on these advantages, 5G networks have introduced crosshaul as an innovative architecture design, which aims at a smooth integration of heterogeneous fronthaul and backhaul technologies and services. The crosshaul enables a flexible and software-defined reconfiguration of all networking elements in a unified hauling environment [2]. As specified by the 5G-Crosshaul project in the European Union's Horizon 2020 Programme [3], crosshaul interconnects mobile edge nodes (e.g., remote radio heads, 5G points of attachment (5GPoA), and edge servers) in the eastbound computations and points-of-presence of the core networks and services in the westbound computations. As a result, high-capacity transmission links and boundary computations are delivered, characterizing the 5G crosshaul.

Despite these supportive features, stabilization, energy efficiency, and latency have not been jointly considered to optimize the crosshaul performance, especially for dealing with the rapid emergence of dense IoT systems. The stability indicates an ability of the system in handling internal parameters against the change of external impacts in order to maintain computation offloading service available. In particular, either local task execution in individual IoT devices or external computation offloading to the eastbound computing is a big challenge owing to resource constraints. Further, remote computation offloading to westbound computing platforms results in time and energy consumption problems, which are not acceptable for time-sensitive and green IoT applications. For instance, smart manufacturing systems, in which more than 10,000 IoT devices are deployed to a dense factory site, are expected to continuously generate dozens of gigabytes per second of traffic volume into the network. The huge traffic overloads any typical edge server in the eastbound platforms. From another perspective, the traffic consumes extremely large amounts of transmission resources on the crosshaul links as well as time and energy resources of the cloud in the westbound platforms [4]. In the context of such big IoT data paradigm, a flexible orchestration between eastbound and westbound computations of the crosshaul is necessary to serve the user applications with their diverse demands.

A. Literature Review

This literature review focuses on the existing schemes [5]–[14] that handle the offloaded data from the IoT devices to the network. These schemes can be classified into three categories: energy efficiency [5]–[8], quality of services (QoS) [9]–[11], and offloading latency [12]–[14]

Approaches aiming at energy efficiency [5]–[8] consider the energy consumption of offloaded data execution as their objective function. For instance, Mao *et al.* [5] proposed a dynamic computation offload assignment policy, which is based on the task failure metric to minimize the CPU-frequency utilization of edge servers and transmit power for data offloading in the eastbound computations. Dinh *et al.* [6] proposed task allocation and computational CPU frequency-scaling techniques to minimize the energy consumption for offloading operation among edge/cloud servers. In addition, several approaches that resolve the joint optimization problem of energy efficiency

The authors are with the School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea (e-mail: dnngoc@uclab.re.kr; dnvu@uclab.re.kr; joongheon@cau.ac.kr; wsna@uclab.re.kr; srcho@cau.ac.kr).

This work was supported by the Space Core Technology Development Program through the National Research Foundation of Korea (No. NRF-2018M1A3A3A02066018).

Manuscript received March 27, 2018; revised September 15, 2018.

Corresponding author: Sungrae Cho.

and computing latency have been proposed by exploiting specific characteristics of their applying scenarios [7], [8]. Nevertheless, the above-mentioned approaches locally target on either the eastbound or westbound computations separately.

QoS-based approaches [9]–[11] target at providing a maximum number of massive IoT devices in obtaining a fair chance to utilize the crosshaul computing services, i.e., offloading serviceability. In [9], Dao *et al.* proposed an adaptive resource balancing scheme, which utilizes the Hungarian method and backpressure algorithm to balance and maximize networking serviceability in the eastbound computations. Quality-aware traffic offloading approach [10] provides an incentive mechanism to motivate edge/cloud servers with better computing power to support lower-computing-capable servers to obtain offload balance among the servers as well as to maximize offload service availability. In [11], cache size in 5GPoA is considered as the representative of system stability to develop a stochastic decision-making algorithm for distributing offloaded data in the eastbound computations. Despite their achievable merits, the aforementioned schemes are incomprehensive solutions as they optimize the offloading services only for IoT device satisfaction. Therefore, resource utilization inside edge/cloud or crosshaul computations might not be sufficiently considered and suffers from inefficiency problems. Particularly, the severity of the problem intensifies for big data in the dense IoT networks.

In latency-oriented approaches [12]–[14], offloaded data-assignment optimization among edge/cloud servers is one of the most important targets. For instance, the index-based task assignment policy [12] minimizes the waiting and execution times of workloads by modeling an edge/cloud server by using the discrete-time Markov decision process. The approach is transformed into index policies for optimal task assignment and power-delay tradeoff. In [13], various IoT applications of offloaded data were considered to select appropriate cloudlet services for reducing the application execution latency and computation costs. In addition, machine learning can be utilized to identify the offloading patterns [14]. Based on the patterns, an optimal task assignment is precalculated to provide approximate optimal online task scheduling in an eastbound computing platform. However, the shortcoming of these schemes is that for achieving minimal latency, the networks possibly undergo energy consumption overheads. Furthermore, the cloud suffers from inefficient utilization due to high latency. This leads to imbalance in the crosshaul computing framework.

Although the literature comprises many studies regarding upstream-data offloading optimization, most previous techniques have focused on separate computing domains, i.e., mobile edge computing in the eastbound platforms or remote cloud computing in the westbound platforms. One-side computing optimizations are inappropriate against the recent emergence of the big data paradigm in dense IoT networks, in which the heterogeneity and massiveness of IoT applications still remain an open challenge. By exploiting the merits of crosshaul infrastructure, we propose a stabilized green crosshaul orchestration (SGCO), which harmonizes the computation capabilities of mobile edges and cloud to provide a green computational framework in both the eastbound and westbound platforms by strictly considering system stability and characteristics of IoT services (such as workload complexity, size, and latency threshold).

B. Main Contributions

The main contributions of this paper are summarized as follows.

- SGCO is proposed as a novel crosshaul computing orchestration; it aims at time-average minimization of energy consumption in upstream data offloading over the crosshaul network. As the cache buffer is stabilized and the computation powers are controlled, the SGCO provides adjustable computing latency threshold for various IoT services.
- From a systematic perspective, the crosshaul computing orchestration is considered as a two-tier queuing system representing the computation capabilities of both eastbound and westbound computations, respectively. During each time unit, the data bifurcation to the eastbound and westbound is decided based on a balancing consideration of system stability, energy consumption, and workload execution complexity.
- From a technological perspective, the adaptive computation optimization of SGCO is developed using the *Lyapunov theory*. Depending on the current cache buffer size and offloaded data rate, the optimal amount of processing data as well as CPU frequencies at the eastbound and westbound are calculated following the *drift-plus-penalty (DPP) expression* policy. Furthermore, for the amount of workload transmitted over the crosshaul links, a data reduction technique is applied by considering the workload execution complexity through a *binary min-knapsack problem* (minKP).
- From an analytical perspective, we analyzed the verification of the SGCO scheme, which possesses a low complexity in both the time and space domains. Moreover, the simulation results show that the SGCO scheme achieves significant performance improvements in terms of energy consumption as well as cache buffer stability.

II. Crosshaul Computing Framework

In this section, we investigate the SGCO crosshaul computing model adopting the 5G-Crosshaul architecture [2], which harmonizes both the eastbound and westbound computations for the upstream data offloading process. The eastbound computation includes cache buffers and edge servers located at the 5GPoA in proximity to user devices, while the westbound computation includes the cloud located at the Internet/cloud service providers. Crosshaul links interface between the eastbound and westbound computations, as depicted in Fig. 1. The bottom portion of Fig. 1 shows the network model of layered connections among participants (i.e., IoT devices, edges, and cloud) in the entire network, while the top portion shows an abstraction of data processing and transfer harmonized by the SGCO crosshaul computing framework. Referred to the 5G-Crosshaul architecture, the SGCO schemes should

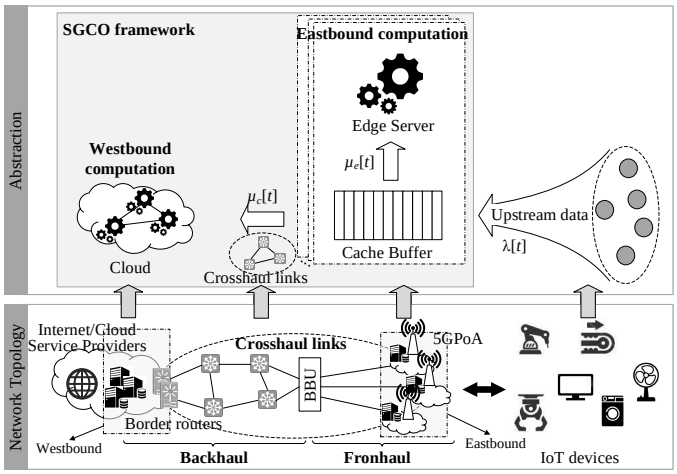

Fig. 1. SGCO computing framework for upstream IoT data offloading.

be implemented in the virtual infrastructure manager (VIM) entities to handle computing resources of the crosshaul.

A. Computational Model

In terms of computation offloading, a workload can be characterized by a tuple of three parameters: its own upstream data size in bits, computational complexity in cycles/bit, and response data size in bits obtained as a result of the workload computation; these are respectively denoted as $\langle u_i, c_i, r_i \rangle$ for the *i-th* workload. Among these parameters, the upstream data size u_i of a workload is considered to be equal to the total workload size if the transmission used is file-transfer type; otherwise, it is considered to be equal to the data block size if the transmission used is stream-transfer type. The workload computation complexity c_i, which varies depending on the application, can be predetermined based on several discrete levels through practical evaluation and classification in offline analysis [15].

In this paper, the energy in Joules consumed by a delegated computing server (i.e., edge server or the cloud) during one computing cycle adopts the widely accepted model of $\kappa f^2[t]$ [16], where κ is the energy coefficient factor specified for each particular CPU reference architecture and $f[t]$ is the CPU frequency at unit time t. Accordingly, the energy consumption during unit time t for workload execution at the edge server and cloud are given by

$$E_e[t] = \kappa_e f_e^2[t]\mu_e[t], 0 \le \mu_e[t] \le f_e[t] \le F_e \text{ and} \quad (1)$$

$$E_c[t] = \kappa_c f_c^2[t]\mu_c[t], 0 \le \mu_c[t] \le f_c[t] < \infty, \text{ respectively,} \quad (2)$$

where F_e is the maximum CPU frequency supported by the edge server. $\mu_e[t]$ and $\mu_c[t]$ are the virtual offloaded data rates in the computing cycle unit assigned to the edge server and cloud during unit time t, respectively. The virtual offloaded data rate is the summation of multiplications between the upstream data rate and computational complexity of each workload. We assume that the cloud, which provides adjustable CPU frequency, adapts to the amount of virtual offloaded data rate by using the dynamic voltage and frequency scaling (DVFS) technique [17]. Moreover, the CPU frequency adjustability, if possible, is considered as an advanced feature in the edge sever [5].

B. Cache Buffer

To ensure generality, we independently consider upstream traffic as a stochastic process. The access uplink from IoT devices to the 5GPoA has a permanent bandwidth of ω for achieving an upstream data rate of $\gamma[t]$ during unit time t. By using the Shannon–Hartley formula, we derive the maximum upstream data rate Γ that might arrive at the 5GPoA during each unit time as $\omega \log(1 + \text{SINR}_{\max})$, where $\text{SINR}_{\max}$ is the maximum signal-to-interference-plus-noise ratio of the IoT devices associated with the 5GPoA. Although $\gamma[t]$ is an unpredictable parameter, Γ is a constant upper-bounded threshold. The maximum value $\text{SINR}_{\max}$ is obtained when an ideal environment condition occurs. In other words, $\text{SINR}_{\max} = \frac{gp}{N_0}$, where g, p, and N_0 are the channel gain, maximum transmit power of the IoT devices, and background noise, respectively. Therefore, a necessary condition for system stability is that the obtained cache buffer should be deployed to satisfy at least a Γ bit-size during each unit time.

There are $N[t]$ workloads in workload set $\mathcal{N}[t]$, which reached the cache buffer during unit time t. Thus, $\gamma[t]$ can also be derived from the summation of all upstream data sizes of the workload $\sum_{i=0}^{N[t]} u_i$ obtained during unit time t. Therefore, the virtual upstream data rate $\lambda[t]$ in the computing cycle unit during unit time t is given by

$$\lambda[t] = \sum_{i=0}^{N[t]} u_i \times c_i. \quad (3)$$

Accordingly, the current virtual cache buffer size $b[t]$ in the computing cycle for the 5GPoA at unit time t is obtained as

$$b[t] = b[t-1]+\lambda[t-1]-(\mu_e[t-1]+\mu_c[t-1]),\ t = 1, 2, 3, \ldots \quad (4)$$

and b[0] = 0 at the initial time point. It is true that the virtual offloaded data rates to the edge server and cloud cannot exceed the total amount of virtual upstream data rate and cached data at the buffer from the previous time unit. Therefore, $\mu_e[t] + \mu_c[t] \le b[t-1] + \lambda[t], \forall t$, and thus $b[t] \ge 0, \forall t$.

C. Crosshaul Transmission

The crosshaul links interconnect the eastbound and westbound computations. We denote the available crosshaul bandwidth at timeslot t by BW[t]. In addition, we assume that the offloaded data to the cloud can be immediately transmitted over the crosshaul links without any scheduling latency [2], [3], [18]–[20].

III. Problem Formulation

A. Workload Execution in SGCO

Considering the SGCO softwarization at unit time t, the cumulative energy consumption during $[0, t]$ is given by $\sum_{\tau=0}^{t}(E_e[\tau] + E_c[\tau])$. It is true that minimizing the energy

consumption during $[0, t]$ is equivalent to minimizing the time-average energy consumption during that time. In other words, we have

$$\min : \sum_{\tau=0}^{t}(E_e[\tau] + E_c[\tau]) \equiv \min : \frac{1}{t}\sum_{\tau=0}^{t}(E_e[\tau] + E_c[\tau]). \quad (5)$$

With respect to the time and space cache buffer stability, the energy-efficient SGCO problem in the infinite timing scale can be generalized as

$$(\mathcal{F}_1) \quad \min_{f_e[t], f_c[t], \mu_e[t], \mu_c[t]} : \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t}(E_e[\tau] + E_c[\tau]) \quad (6)$$

$$\text{s.t. (1), (2),}$$

$$\lim_{t\to\infty} \frac{b[t]}{t} < \infty. \quad (7)$$

Constraint (7) ensures that the cache buffer is stabilized in both the time and space domains.

B. Crosshaul Transmission

Given that $\mu_c^*[t]$ is the optimal solution of the virtual offloaded data rate assigned to the cloud at unit time t, which is derived from minimizing function $\mathcal{F}_1$, $\mu_c^*[t]$ should be transferred to the cloud over the crosshaul links for execution. However, $\mu_c^*[t]$ is measured in terms of computing cycle units, which is inappropriate for measuring transmission cost. In addition, the transmission costs of energy and latency increase proportionally with the bit-sized amount of data transmitted. The set and number of workloads that should to be transferred to the cloud in unit time t are denoted by $\mathcal{N}_c[t]$ and $N_c[t]$, respectively. Considering the properties of both the upstream and response data, the transmission cost minimization problem can be expressed as

$$(\mathcal{F}_2) \quad \min_{\mathcal{N}_c[t]} : \sum_{i=0}^{N_c[t]} (u_i + r_i) \quad (8)$$

$$\text{s.t. } \mathcal{N}_c[t] \in \mathcal{N}[t], \quad (9)$$

$$\sum_{i=0}^{N_c[t]} (u_i \times c_i) = \mu_c^*[t] + \epsilon[t], \quad \epsilon \in \mathbb{Z}, \quad (10)$$

$$|\epsilon[t]| < \max\{u_i \times c_i | \forall i = 1, 2, \ldots, N[t]\}. \quad (11)$$

Constraints (10) and (11) ensure that the total computing cycle required for the selected workloads is close to the optimal virtual offloaded traffic.

IV. Crosshaul Computation Optimization

A. Joint Optimization Transformation

Consider the SGCO softwarization depicted in Fig. 1; the cache buffer in the 5GPoA operates as a queuing system, which is characterized by an arrival rate of approximately $\lambda[t]$ and a departure rate of approximately $(\mu_e[t]+\mu_c[t])$ at timeslot t. Recall that $\lambda[t]$, $\mu_e[t]$, and $\mu_c[t]$ are measured in computing cycle units. Let $\mathbb{B}[t]$ be a state vector of the cache buffer at timeslot t, which is defined by

$$\mathbb{B}[t] \triangleq \{b[\tau] | \tau = 1, 2, 3, \ldots, t\}. \quad (12)$$

Following the Lyapunov theory [21], the quadratic Lyapunov function $L[t]$, which represents a scalar measure of the cache buffer size during $[0, t]$, is given by

$$L[t] \triangleq \frac{1}{2}\mathbb{B}^\top[t]\mathbb{B}[t] = \frac{1}{2}\sum_{\tau=1}^{t} b^2[\tau], \quad (13)$$

where $\mathbb{B}^\top[t]$ is the transpose vector of $\mathbb{B}[t]$. The change of $L[t]$ from timeslot t to $t + 1$ is defined by

$$\Delta L[t] \triangleq \mathbf{E}(L[t + 1]|\mathbb{B}[t]) - L[t], \quad (14)$$

which is known to be the Lyapunov drift at timeslot t, where $\mathbf{E}(\cdot)$ is a conditional expectation. The conditional expected Lyapunov drift $\mathbf{E}(\Delta L[t]|\mathbb{B}[t])$ has been proven to be bounded above as follows [22]:

$$\mathbf{E}(\Delta L[t]|\mathbb{B}[t]) \leq C + b[t](\lambda[t] - (\mu_e[t] + \mu_c[t])), \quad (15)$$

where C is a determinable finite constant value. Therefore, we achieve cache buffer stabilization when minimizing $b[t](\lambda[t] - (\mu_e[t] + \mu_c[t]))$.

Further, the goal of optimization problem $\mathcal{F}_1$ is to accommodate IoT devices with a minimal computation energy consumption in online situations while maintaining the cache buffer stabilized. Fortunately, an (approximately) optimal solution can be achieved by greedily minimizing the bound on the corresponding DPP policy $\mathcal{P}_1$ at each timeslot t [21], which is defined by

$$\mathcal{P}_1 \triangleq V(E_e[t] + E_c[t]) + b[t](\lambda[t] - (\mu_e[t] + \mu_c[t])), \quad (16)$$

where V is the positive control factor. In policy $\mathcal{P}_1$, $(E_e[t] + E_c[t])$ is the objective function (referred to as the *penalty*) and $b[t](\lambda[t]-(\mu_e[t]+\mu_c[t]))$ represents the key constraint of cache buffer stabilization (named as the *drift*). Finally, optimization problem $\mathcal{F}_1$ can be transformed into a joint optimization problem $\mathcal{F}_3$ as follows:

$$(\mathcal{F}_3) \quad \min : \mathcal{P}_1 \quad (17)$$

$$\text{s.t. (1), (2), (7).}$$

Moreover, we show that the optimal value of the controllable CPU frequency and virtual offloaded data rate are approximate in each timeslot by using the following theorem.

Theorem 1: Given the virtual offloaded data rate $\mu[t]$ in computing cycles at timeslot t, the optimal value of the controllable CPU frequency $f^*[t]$ required for a computing agent to achieve the minimum energy consumption $E^*[t]$ for an execution of $\mu[t]$ is equal to $\mu[t]$.

Proof: From (1) and (2), $E^*[t] = \kappa(f^*[t])^2\mu[t]$. Following the constraint $f[t] \geq \mu[t]$, a necessary and efficient condition for the achievable minimum value $E^*[t]$ of the energy consumption is that the controllable CPU frequency is calibrated down to its minimum value, i.e., $f^*[t] = \mu[t]$. As a result, $E^*[t] = \kappa(f^*)^3[t]$. ■

B. Case 1: Uncontrollable-CPU-Frequency Edge Server

Condition expression (C_1): As the cloud can adapt the CPU frequency according to the amount of offloaded traffic, the condition *uncontrollable-CPU-frequency edge server* leads to

$$(C_1) \quad \begin{cases} f_e[t] = F_e, \\ f_c[t] = \mu_c[t], \end{cases} \quad \forall t. \quad (18)$$

Within condition C_1, optimization problem $\mathcal{F}_3$ is updated as

$$(\mathcal{F}_{3-1}) \quad \min_{\mu_e[t], f_c[t]} : V(\kappa_e F_e^2 \mu_e[t] + \kappa_c f_c^3[t]) + b[t](\lambda[t] - (\mu_e[t] + f_c[t])) \tag{19}$$

$$\text{s.t. } \mu_e[t] + f_c[t] \le b[t] + \lambda[t], \tag{20}$$

$$b[0] = 0, \tag{21}$$

$$0 \le \mu_e[t] \le F_e, \tag{22}$$

$$0 \le f_c[t] \le \mathrm{BW}[t], \tag{23}$$

$$\mu_e[t], f_c[t] \in \mathbb{N}. \tag{24}$$

In $\mathcal{F}_{3-1}$, the objective function is used to minimize the summation of a linear polynomial function of $\mu_e[t]$ and a cubic polynomial function of $f_c[t]$, while the constraints adopt the *Karush–Kuhn–Tucker (KKT)* conditions. The detailed solving process for this problem is described in [23], and has thus been omitted from the scope of this paper. The optimal amount of traffic offloaded to the edge server is denoted by $\mu_e^*[t]$. The conditionally optimal solutions are provided as follows:

$$f_c^*[t] = \min\left\{\sqrt{\frac{b[t]}{3V\kappa_c}}, b[t] + \lambda[t], \mathrm{BW}[t]\right\}, \tag{25}$$

$$\mu_e^*[t] = \arg\min_{\varphi_1, \varphi_2}\{\mathcal{F}_{3-1}\}, \tag{26}$$

where

$$\varphi_1 \triangleq \begin{cases} \mu_e^*[t] = \sqrt{\frac{b[t]}{3V\kappa_c}} & \text{, if } V\kappa_e F_e^2 \ge b[t] \ \&\ F_e \ge \sqrt{\frac{b[t]}{3V\kappa_c}} \\ \mu_e^*[t] = F_e & \text{, if } V\kappa_e F_e^2 < b[t] \ \&\ F_e \ge \sqrt{\frac{b[t]}{3V\kappa_c}} \end{cases} \tag{27}$$

$$\varphi_2 \triangleq \begin{cases} \mu_e^*[t] = 0 & \text{, if } V\kappa_e F_e^2 \ge b[t] \\ \mu_e^*[t] = \min\left\{F_e, \sqrt{\frac{b[t]}{3V\kappa_c}}\right\} & \text{, otherwise.} \end{cases} \tag{28}$$

C. Case 2: Controllable-CPU-Frequency Edge Server

Condition expression (C_2): Both the edge server and cloud can adapt the CPU frequency according to the amount of offloaded traffic, and therefore the condition *controllable-CPU-frequency edge server* is represented by

$$(C_2) \quad \begin{cases} f_e[t] = \mu_e[t], \\ f_c[t] = \mu_c[t], \end{cases} \quad \forall t. \tag{29}$$

Condition C_2 for optimization problem $\mathcal{F}_3$ leads to the closed form expression given as

$$(\mathcal{F}_{3-2}) \quad \min_{f_e[t], f_c[t]} : V(\kappa_e f_e^3[t] + \kappa_c f_c^3[t]) + b[t](\lambda[t] - (f_e[t] + f_c[t])) \tag{30}$$

$$\text{s.t. (21), (23), (24),}$$

$$f_e[t] + f_c[t] \le b[t] + \lambda[t], \tag{31}$$

$$0 \le f_e[t] \le F_e. \tag{32}$$

The objective function $\mathcal{F}_{3-2}$ is used to minimize the summation of the two cubic polynomial functions of $f_e[t]$ and $f_c[t]$. Within the same constraint characteristics as that of $\mathcal{F}_{3-1}$, problem $\mathcal{F}_{3-2}$ can be resolved by the conditionally optimal solutions as follows:

$$\{f_e^*[t], f_c^*[t]\} = \arg\min_{\omega_1, \omega_2, \omega_3}\{\mathcal{F}_{3-2}\}, \tag{33}$$

where

$$\omega_1 \triangleq \begin{cases} f_e^*[t] = 0, \\ f_c^*[t] = 0, \end{cases} \tag{34}$$

$$\omega_2 \triangleq \begin{cases} f_c^*[t] = \min\left\{b[t] + \lambda[t] - \sqrt{\frac{b[t]}{3V\kappa_e}}, \sqrt{\frac{b[t]}{3V\kappa_c}}, \mathrm{BW}[t]\right\}, \\ f_e^*[t] = \min\left\{\sqrt{\frac{b[t]}{3V\kappa_e}}, F_e\right\}, \\ b[t] + \lambda[t] \ge \sqrt{\frac{b[t]}{3V\kappa_c}}, \end{cases} \tag{35}$$

$$\omega_3 \triangleq \begin{cases} \begin{cases} f_e^*[t] = \min\{F_e, b[t] + \lambda[t] - f_c^*[t]\}, \\ f_c^*[t] = \min\{b[t] + \lambda[t], \psi, \mathrm{BW}[t]\}, \\ \psi = \max\left\{\frac{9V^2\kappa_e\kappa_c(b[t]+\lambda[t])^2 - 3V\kappa_e(b[t]+\lambda[t])}{3V(\kappa_c - \kappa_e)}, 0\right\}, \\ \kappa_e \le \kappa_c, \end{cases} \\ \begin{cases} f_e^*[t] = \min\{F_e, b[t] + \lambda[t] - f_c^*[t]\}, \\ f_c^*[t] = \min\left\{\arg\min_{0, b[t]+\lambda[t]} \mathcal{F}_{3-2}, \mathrm{BW}[t]\right\}, \\ \kappa_e > \kappa_c. \end{cases} \end{cases} \tag{36}$$

Algorithm 1 Crosshaul Computation Optimization.

- **Input:**
  $b[0] = 0$;
  Observe $\lambda[t]$;
- **Output:**
  $f_e^*[t], \mu_e^*[t], f_c^*[t], \mu_c^*[t]$;

Repeat each timeslot t

Case 1: $\mathbf{f_e^*[t] = F_e}$ ***and*** $\mathbf{f_c^*[t] = \mu_c^*[t]}$

Calculate $\sqrt{\frac{b[t]}{3V\kappa_c}}$, $b[t] + \lambda[t]$, and $\sqrt{\frac{V\kappa_e F_e^2}{3\kappa_c}}$;

$\mu_e^*[t]$ and $f_c^*[t]$ are given by Eq. 25;

$b[t] = b[t] + \lambda[t] - (\mu_e^*[t] + \mu_c^*[t])$;

Case 2: $\mathbf{f_e^*[t] = \mu_e^*[t]}$ ***and*** $\mathbf{f_c^*[t] = \mu_c^*[t]}$

Calculate $\sqrt{\frac{b[t]}{3V\kappa_c}}$, $\sqrt{\frac{b[t]}{3V\kappa_e}}$, $b[t] + \lambda[t]$, and ψ;

$f_e^*[t]$ and $f_c^*[t]$ are given by Eq. 33;

$b[t] = b[t-1] + \lambda[t] - (\mu_e^*[t] + \mu_c^*[t])$;

End

The corresponding pseudo-code for the crosshaul computation optimization is presented in Algorithm 1. Optimal frequencies $f_e^*[t]$ and $f_c^*[t]$ for CPU adjustments are determined in the edge server and cloud, respectively. Moreover, the amounts of offloaded traffic in the computing cycle unit assigned for the edge server and cloud are $\mu_e^*[t]$ and $\mu_c^*[t]$, respectively.

V. Crosshaul Transmission Optimization

As analyzed in Section III-B, the goal of optimization function $\mathcal{F}_2$ is to minimize the amount of offloaded traffic in the bit-size unit from the cache buffer at the 5GPoA via the crosshaul links to the cloud. It is assumed that the cloud has an input memory for buffering the incoming traffic, wherein the possible overhead of the incoming traffic can be scheduled when $\epsilon[t] \geq 0$. Therefore, constraints (10) and (11) in $\mathcal{F}_2$ can be relaxed by a lower bound of the total computing cycle $\mu_c^*[t]$ at timeslot t. Consequently, the minimization problem can be transformed into the following function:

$$(\mathcal{F}_4) \quad \min : \sum_{i=0}^{N[t]} x_i(u_i + r_i) \tag{37}$$

$$\text{s.t. } x_i \in \{0, 1\}, \tag{38}$$

$$\sum_{i=0}^{N[t]} x_i(u_i \times c_i) \geq \mu_c^*[t] - \underbrace{\left(\sum_{j=0}^{N[t-1]} x_j(u_j \times c_j) - (\mu_c^*[t-1] - \delta[t-1]) \right)}_{\delta[t]}, \tag{39}$$

where $N[0] = 0$, $\mu_c^*[0] = 0$, and $\delta[0] = 0$. In constraint (39), $\delta[t]$ represents the remaining traffic scheduled in the previous timeslot $t - 1$.

Theorem 2: The possible overhead of the incoming traffic is less than the maximum computing cycles required by a workload in the incoming workload set.

Proof: The reductio ad absurdum method is used to prove Theorem 2. $\mathcal{N}_c[t]$ is the optimal set of workloads satisfying minimization function $\mathcal{F}_4$. If the overhead of the incoming traffic is greater than or equal to the maximum computing cycles required by a workload in $\mathcal{N}[t]$, we have

$$\sum_{i=0}^{N_c[t]} x_i(u_i \times c_i) - (\mu_c^*[t] - \delta[t]) \geq \max\{u_i \times c_i | \forall i \in \mathcal{N}[t]\}. \tag{40}$$

However, when we drop an arbitrary workload, e.g., the *k-th* workload from $\mathcal{N}_c[t]$,

$$\sum_{i=0}^{N_c[t]\setminus\{k\}} x_i(u_i \times c_i) - (\mu_c^*[t] - \delta[t]) \geq \underbrace{\geq \max\{u_i \times c_i | \forall i \in \mathcal{N}[t]\} - (u_k \times c_k)}_{\geq 0, \forall k}. \tag{41}$$

Therefore, the set of workload $\mathcal{N}_c[t] \setminus \{k\}$ still satisfies constraint (39). This implies that $\mathcal{N}_c[t]$ is not the optimal set of workloads for minimization function $\mathcal{F}_4$. In other words, the overhead of the incoming traffic cannot be greater than or equal to the maximum computing cycles required by a workload in the incoming workload set. ■

From another perspective, it is recognized that function $\mathcal{F}_4$ is a minKP problem [24], in which the workloads are selected to achieve minimal bit-sized traffic while maintaining a computing cycle unit of at least $\mu_c^*[t] - \delta[t]$. In relevant literature, a variety of solutions have been proposed to solve such a minimization problem, including both exact and approximate solutions [25]. However, as minKP is considered as an NP-hard problem, the computational complexity of existing approaches still remains an open challenge, which may not be appropriate for application in resource-constrained equipment such as 5GPoA. Moreover, $(u_k \times c_k) \ll \sum_{i=0}^{N_c[t]} x_i(u_i \times c_i), \forall k \in N_c[t]$ in massive IoT environments. Hence, the minimization problem $\mathcal{F}_4$ can be resolved through an approximate solution by using a greedy heuristic algorithm, in which the incoming workloads are sorted in the descending order to select the maximum-required-computing-cycle workloads, assuming constraint (39) and Theorem 2 are satisfied. The corresponding scheme is presented in Algorithm 2.

Algorithm 2 Crosshaul Transmission Optimization.

- **Input:** $\mathcal{N}[t]$
- **Output:** $\mathcal{N}_c[t]$

Function DESCENDING($\mathcal{N}[t]$) by $\{u_i \times c_i\}$ indexing;
for $(i = 0, i < |\mathcal{N}[t]|, i++)$ **do**
  $\mathcal{N}_c[t] \leftarrow \{i | i \in \mathcal{N}[t]\}$;
  if Constraint (39) is true **then**
    Break;
End

VI. Computational Complexity Analysis

As expressed in Section IV, the minimization problem $\mathcal{F}_{3-1}$ is a combination of cubic function $f_c[t]$ and linear function $\mu_e[t]$, whereas the minimization problem $\mathcal{F}_{3-2}$ is in the form of two cubic functions $f_e[t]$ and $f_c[t]$. As depicted in Algorithm 1, the conditionally optimal solutions for these two problems are calculated based on the given parameters (i.e., V, κ_e, κ_c, and F_e) and observed parameters (i.e., $b[t]$ and $\lambda[t]$) by using normal operations. Therefore, the time complexity of the solutions can be identified as $O(1)$. Moreover, since Algorithm 1 iterates its operations in each timeslot, the intermediate values are overwritten in its occupied memories for each iteration. Hence, the space complexity is $O(1)$.

Additionally, although the crosshaul transmission optimization $\mathcal{F}_4$ is considered to be a min-KP problem, as stated in Section V, an approximate solution can be achieved using a simple sorting algorithm in the descending order (see Algorithm 2). Based on the method in [26], the array-sorting algorithm might introduce time and space complexities of $O(n \log(n))$ and $O(1)$, respectively, where n is the number of workloads in the cache buffer during one timeslot. In summary, the proposed SGCO scheme supports time and space complexities of $O(n \log(n))$ and $O(1)$, respectively.

VII. Performance Evaluation

A. Simulation Settings

Simulation model and initial parameters: For the simulations, we used a network model, as depicted in Fig. 1. The offloaded data parameters generated from the IoT devices are deployed as follows. During each timeslot unit, there is an arbitrary number of IoT devices in the range of [500, 1000]

that require computation offloading services, and each IoT device transmits a workload within a size of 50–100 KB to the network for processing, representing various services such as environmental sensor reading, auto machinery, and navigation systems. In this context, given a computational complexity set of $\{10, 20, 50, 100\}$ cycle/bit, which we can obtain through practical experiences and classifications, as in [27], each workload is mapped to a computational complexity in this set. After execution, the SGCO returns the response data of each workload to the corresponding IoT device. The maximum CPU frequency of the edge server is setup equal to (100 + 10)% of the average virtual upstream data rate $\mu_e[t]$ that is handled by the edge with 10% redundancy. On the contrary, the CPU at the cloud is flexibly adjusted to be equal to the current data rate $\mu_c[t]$ that is offloaded to the cloud owing to the virtualization ability. Without loss of generality, the energy coefficient factors κ_e and κ_c are assumed to be 5×10^{-25} and 4×10^{-25} (owing to the virtualization ability) [27], respectively.

Competitor definition: For performance comparison, we additionally considered five benchmark schemes including *energy-efficient edge server optimization (EE), arrival-aware offloading scheme (AA), complexity-aware offloading scheme (CA), maximum edge computing exploitation (ME), and remote cloud offloading scheme (RC)* [5], [28], [29]. Detailed descriptions of these schemes are as follows:

- *Energy-efficient edge server optimization (EE) scheme:* The EE scheme minimizes the energy consumption utilized for workload computing by scheduling the CPU core operations at the edge server. The number of activated CPU cores depends on the workload arrival rate.
- *Arrival-aware offloading (AA) scheme:* The AA scheme adapts the CPU frequency according to the average arrival rate of the offloaded data at the network.
- *Complexity-aware offloading (CA) scheme:* The arrived workloads are sorted in the ascending order depending on their computational complexities. To minimize the execution latency, the CA scheme prioritizes the lowest-computational-complexity workload to be executed as much as the maximum capacity of the edge server. The remaining workloads are forwarded to the cloud for remote processing.
- *Maximum edge computing exploitation (ME) scheme:* To reduce the execution waiting latency, the arrived workloads are greedily computed at the edge server online, and the remaining workloads are continuously forwarded to the cloud for remote processing.
- *Remote cloud offloading (RC) scheme:* This is the conventional cloud computing strategy, i.e., the total arrived workloads are delivered to the cloud. It is worth noting that the edge server plays no role in this model.

Simulation methodology: The simulation process consists of the following three steps.

- *Input-data pattern preparation:* Based on the simulation model and initial parameter setups, 300 Monte Carlo experiments of input data from 1000 IoT devices were randomly generated for use in 300 simulation timeslots.

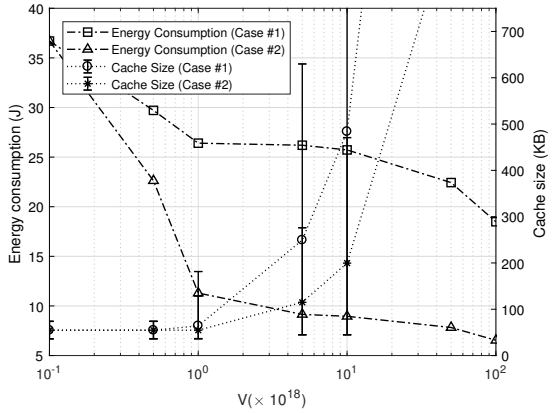

Fig. 2. Effect of factor V on average energy consumption and cache size.

- *System operation:* During each timeslot, an input-data sample was transferred into the system for processing. This step comprised 300 contiguous timeslots. Depending on the evaluation metric, some system parameters were modified accordingly.

B. Effect of the Control Factor V

Fig. 2 shows the effect of V on system stability and energy efficiency. As observed, an increase in V results in a better achievable energy efficiency; however, the system requires a higher cache buffer size. Moreover, the system might face an unstable condition (see the error-bar illustration in Fig. 2). In contrast, a decrease in V prioritizes the system stability; however, the system consumes more energy for workload execution. In particular, the bars show bad results when the control factor V is greater than $10^0 \times 10^{18}$ (for the cache size in Fig. 2) and $10^1 \times 10^{18}$ (for the buffering latency in Fig. 3). These bad results indicate unstability of the system due to bad V selections even though the energy consumption is reduced. Derived from Fig. 2, it is observed that a good selection of V is around $10^0 \times 10^{18}$, which can provide not only low energy consumption but also stability for the system (illustrated by short bars).

In terms of the latency effect, total latency of workload execution by using the SGCO was observed to mostly depend on the buffering latency at the cache in the fronthaul; see Figs. 3 and 4. In Fig. 3, the buffering latency exponentially increases when higher V is used. As the buffering latency has a close relation to the cache size, Figs. 2 and 3 reveal the same effect for both the metrics against the adjustment of V. On the other hand, Fig. 4 shows the effect of V on computing latency. It is observed that the computing latency is not monotone as a function of V because the computing latency is determined by the amount of data processed at the edge $\mu_e[t]$ and the data delivered to and processed at the cloud $\mu_c[t]$. According to Equations (25), (26), and (33), these data is not monotone as a function of V. For each system statement, there is a good range of V factor, by which the joint optimization function (17) obtains balance solution. Especially in this case, V is found to be around $10^0 \times 10^{18}$ resulting in a stable and energy-efficient condition of the system. This value of V is consistent as shown in all Figs. 3, 4, and 4. Note that the crosshaul link between the edge and cloud is set to 10 Gbps to calculate

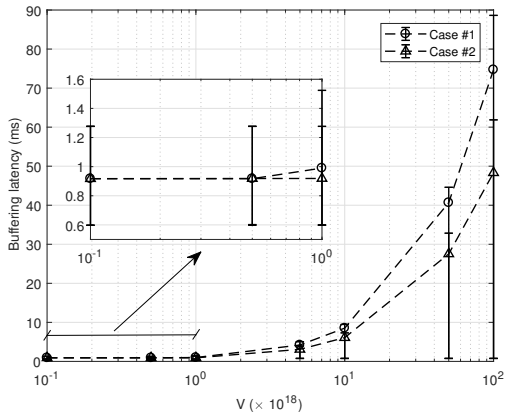

Fig. 3. Effect of factor V on buffering latency.

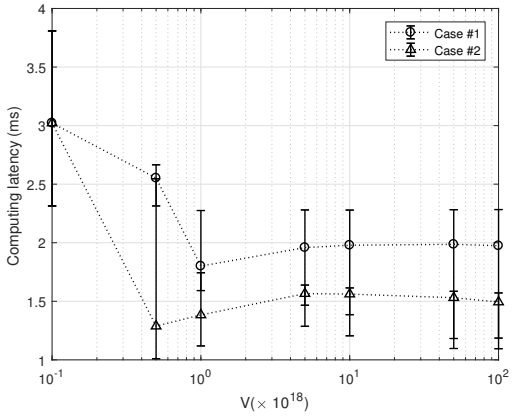

Fig. 4. Effect of factor V on computing latency.

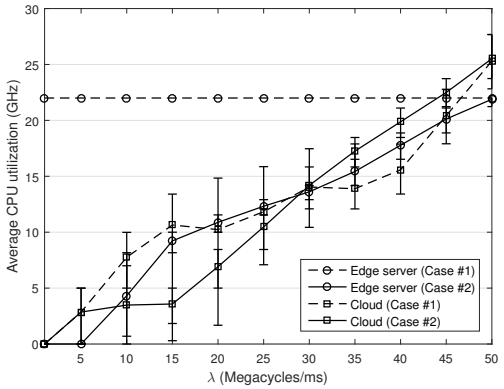

Fig. 5. Average CPU utilization depending on the virtual upstream data rate.

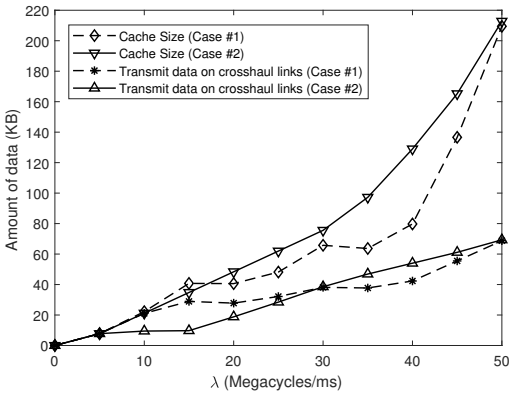

Fig. 6. Data amount buffered at the cache and transmitted on crosshaul links to the cloud depending on the virtual upstream data rate λ.

the crosshaul transmission latency accordingly [18]. With the selected V value of $10^0 \times 10^{18}$, the total latency is averaged to 2.79 ms in the range of [3.80, 2.18] ms (Case #1) and to 2.30 ms in the range of [3.02, 1.72] ms (Case #2).

C. Effect of the Virtual Upstream Data Rate λ

As expressed in Section IV, the optimal CPU frequencies in the edge server and cloud are calculated based on $\lambda[t]$ observations. Fig. 5 illustrates the orchestration of the edge server and cloud adapting to the adjustment of λ. In Case #1, as the CPU frequency in the edge server is uncontrollable, it operates at 22 GHz throughout. In addition, the cloud flexibly assigns the CPU frequency according to the offloaded arrival rate at the cloud. In Case #2, the edge and cloud harmonization generates adaptable CPU frequencies in both the edge server and cloud following objective function $\mathcal{F}_{3-2}$ and Algorithm 1. As observed, even though the average CPU frequencies in the edge server and cloud are increased proportionally to the amount of upstream data, these increases are nonlinear.

Similarly, a higher λ rate leads to higher amounts of data buffered at the cache and transmitted over crosshaul links to the cloud; see Fig. 6. Even though in Case #1, the CPU of the edge server is fixed at 22 GHz, the amount of data buffered at the cache is dynamically managed by using Algorithm 1; and the amount of data transmitted on the crosshaul links is controlled further by using Algorithm 2. For a consistent evaluation, V is maintained at 1×10^{18}. The graph in Fig. 6 reveals that the proposed algorithm exposes its effectiveness when the λ rate reaches below the cache buffer threshold (i.e., 200-KB cache size). When the λ rate exceeds the cache buffer threshold, the speed increases exponentially.

D. Approximation Verification

Table I summarizes the statistic indexes of the paired differences between Algorithm 2 and the minKP method. The comparison was performed using *IBM SPSS Statistics 20* tool in terms of the mean, standard deviation, and standard error mean of the differences over 600 samples for each pair. The measurement units for the cache size and energy consumption per timeslot are in KB and J/ms scales, respectively. The results show that the mean of the difference in cache size between the two algorithms in Cases #1 and #2 are 5.03 and 36.89 bytes, respectively. Moreover, the mean of the difference in energy consumption per timeslot in both the cases is under 0.02 J/ms. Regarding the difference in fluctuation, the small values of standard deviation and standard error mean also prove the approximation between the two algorithms.

The detailed paired-samples statistics of the cache size are summarized in Table II. The means of the cache sizes in Algorithm 2 and the minKP method are approximately 63 and 54 KB in Cases #1 and #2, respectively. The standard deviation demonstrates that the system is more stable in Case #2 than in Case #1. Fig. 7 visualizes the approximation in cumulative energy consumption during 300 timeslots. The

TABLE I
PERFORMANCE COMPARISON BETWEEN ALGORITHM 2 AND THE TYPICAL MINKP METHOD.

	Paired differences			
	#Sample	**Mean**	**Std. deviation**	**Std. error mean**
Pair 1: Alg. 2 vs. minKP in cache size (Case #1)	600	-0.00503	0.03607	0.00208
Pair 2: Alg. 2 vs. minKP in cache size (Case #2)	600	0.03689	0.03931	0.00227
Pair 3: Alg. 2 vs. minKP in energy consumption (Case #1)	600	0.00068	0.04436	0.00256
Pair 4: Alg. 2 vs. minKP in energy consumption (Case #2)	600	-0.01446	0.08410	0.00486

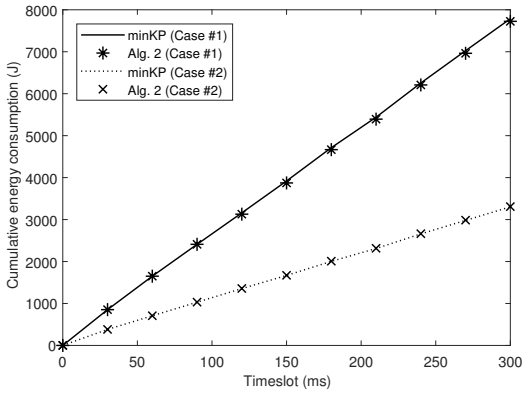

Fig. 7. Performance approximation of cumulative energy consumption between the proposed greedy heuristic algorithm and the typical minKP method.

gaps of cumulative energy consumption between the two verified algorithms are 0.79% and 0.19% in Cases #1 and #2, respectively.

E. Performance Comparison

Cumulative energy consumption evaluation: Figs. 8(a)–(c) represent the cumulative energy consumption at the edge server, cloud, and entire system, respectively. Depending on the algorithm utilized in the system, the energy consumption is distributed between the edge server and cloud based on different strategies. In Fig. 8(a), as the CA and ME schemes utilize the maximum CPU power at the edge server to process the offloaded data, they reveal the highest energy consumption. The SGCO1 scheme, in which the CPU is uncontrollable, shows much smaller energy consumption (approximately 34.35% reduction) owing to its computation harmonization with the cloud. For the schemes in which the CPU adjustment is utilized at the edge server, the SGCO2 and EE schemes demonstrate an approximate performance, which is lower by 24.31% compared to the result of the AA scheme. In contrast, the RC scheme does not spend energy in the edge server.

Fig. 8(b) depicts the energy consumption in the cloud. Compared with the illustration in Fig. 8(a), the RC scheme shown in Fig. 8(b) reveals the highest energy consumption as it executes all the workloads in the cloud only. In contrast, the CA and ME schemes utilize a small energy at the cloud (approximately 13.00 J during 300 ms) while the SGCO1 scheme consumes 2930.32 J in this duration. The SGCO2 scheme has an energy consumption of 987.52 J compared to 1682.64 J and 56.59 J for the EE and AA schemes, respectively.

Average CPU utilization comparison: Fig. 9 demonstrates the flexible orchestration between the edge server and the cloud for handling the incoming offloaded traffic. The CA, ME, and RC schemes were realized to show poor harmonization. That is, either edge server or cloud server is mainly used for the offloading activities. On the other hand, although the maximum CPU frequency is utilized by the SGCO1 scheme, the CPU frequency assigned by the cloud is adapted to the data arrival rate. Among the remaining schemes, the SGCO2 and EE schemes present a balance between the edge server and cloud; it shows the best harmonization to achieve an optimal crosshaul computing. On the contrary, the AA scheme adjusts only the CPU frequency at the edge server according to the arrival rate.

VIII. CONCLUDING REMARKS

The proposed SGCO provides an adaptable offloading computation to distribute the data achieved between the edge server and cloud in the eastbound and westbound computations of the crosshaul, respectively. The amounts of data assigned for the edge server and cloud are dynamically determined according to the recent upstream data rate from IoT devices. The simulation analysis showed that the SGCO provides energy-efficient workload execution in the system while maintaining a stable cache buffer. Moreover, the SGCO reveals a time and space complexities of $O(n\log(n))$ and $O(1)$, respectively. That is, the SGCO is suitable to be implemented in individual 5GPoA to provide self-calibration capability for energy-efficient computation offloading in dense IoT networking environments.

REFERENCES

[1] C. Bu, X. Wang, M. Huang, and K. Li, "SDNFV-based dynamic network function deployment: model and mechanism," *IEEE Communications Letters*, vol. 22, no. 1, pp. 93–96, 2018.

[2] X. Li, R. Casellas, G. Landi, A. de la Oliva, X. Costa-Perez, A. Garcia-Saavedra, T. Deiss, L. Cominardi, and R. Vilalta, "5G-Crosshaul network slicing: Enabling multi-tenancy in mobile transport networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 128–137, 2017.

[3] A. De La Oliva *et al.*, "5G-Crosshaul: The 5G integrated fronthaul/backhaul," (cited Feb. 10, 2018). [Online]. Available: http://5g-crosshaul.eu/wp-content/uploads/2015/11/5G-Crosshaul.pdf

[4] J.-Q. Li, F. R. Yu, G. Deng, C. Luo, Z. Ming, and Q. Yan, "Industrial Internet: A survey on the enabling technologies, applications, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1504–1526, 2017.

[5] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[6] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[7] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2015.

TABLE II
PAIRED SAMPLES STATISTICS OF CACHE SIZE IN KBYTES.

	Algorithm 2		minKP method	
	Case #1	Case #2	Case #1	Case #2
Mean	63.675382	54.365245	63.680417	54.328354
Std. deviation	17.055316	10.277196	17.058823	10.276095
Std. error mean	0.984689	0.593354	0.984892	0.593291

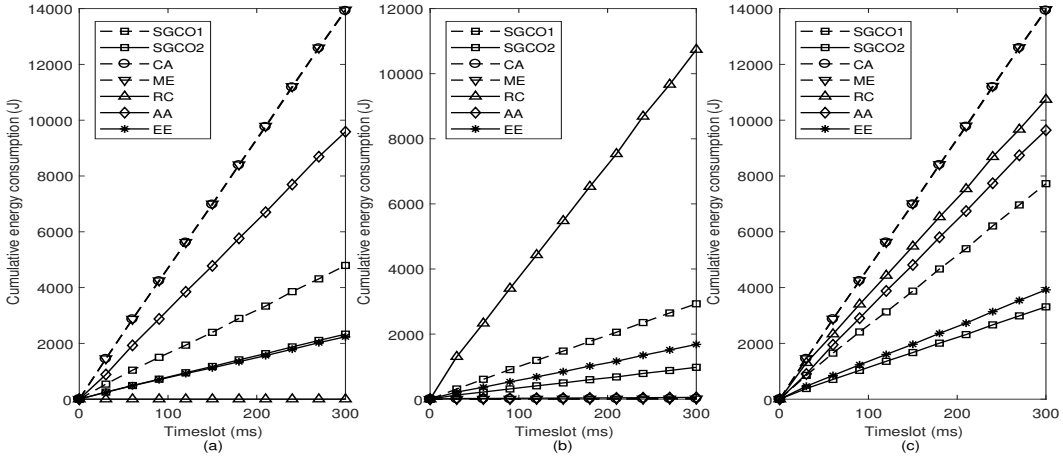

Fig. 8. Cumulative energy consumption at the (a) edge, (b) cloud, and (c) entire crosshaul platform, respectively.

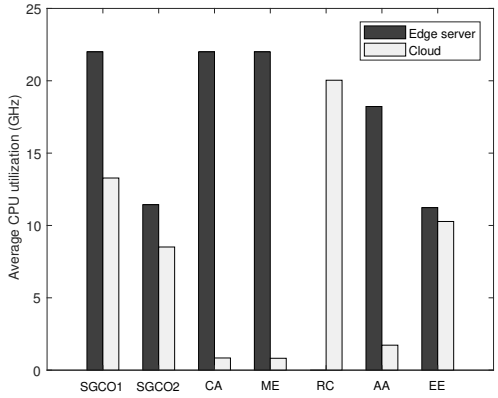

Fig. 9. Average CPU utilization in the edge server and cloud.

[8] J. P. Champati and B. Liang, "Semi-online algorithms for computational task offloading with communication delay," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1189–1201, 2017.

[9] N.-N. Dao, J. Lee, D.-N. Vu, J. Paek, J. Kim, S. Cho, K.-S. Chung, and C. Keum, "Adaptive resource balancing for serviceability maximization in fog radio access networks," *IEEE Access*, vol. 5, pp. 14 548–14 559, 2017.

[10] W. Hu and G. Cao, "Quality-aware traffic offloading in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3182–3195, 2017.

[11] J. Kim and W. Lee, "Stochastic decision making for adaptive crowdsourcing in medical big-data platforms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 11, pp. 1471–1476, 2015.

[12] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *IEEE International Conference on Communications (ICC)*, 2016, pp. 1–7.

[13] D. G. Roy, D. De, A. Mukherjee, and R. Buyya, "Application-aware cloudlet selection for computation offloading in multi-cloudlet environment," *The Journal of Supercomputing*, vol. 73, no. 4, pp. 1672–1690, 2017.

[14] N.-N. Dao, D.-N. Vu, Y. Lee, S. Cho, C. Cho, and H. Kim, "Pattern-identified online task scheduling in multi-tier edge computing for industrial IoT services," *Mobile Information Systems*, vol. 2018, p. 2101206, 2018.

[15] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *USENIX conference on Hot topics in cloud computing (HotCloud)*, 2010, pp. 1–7.

[16] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[17] S. Wang, Z. Qian, J. Yuan, and I. You, "A DVFS based energy-efficient tasks scheduling in a data center," *IEEE Access*, vol. 5, pp. 13 090–13 102, 2017.

[18] X. Costa-Perez, A. Garcia-Saavedra, X. Li, T. Deiss, A. De La Oliva, A. Di Giglio, P. Iovanna, and A. Moored, "5G-Crosshaul: an SDN/NFV integrated fronthaul/backhaul transport network architecture," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 38–45, 2017.

[19] S.-C. Hung, H. Hsu, S.-Y. Lien, and K.-C. Chen, "Architecture harmonization between cloud radio access networks and fog networks," *IEEE Access*, vol. 3, pp. 3019–3034, 2015.

[20] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1818–1831, 2017.

[21] M. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool Publishers, 2010.

[22] J. Kim, G. Caire, and A. F. Molisch, "Quality-aware streaming and scheduling for device-to-device video delivery," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2319–2331, 2016.

[23] Z.-Q. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE Journal on selected areas in communications*, vol. 24, no. 8, pp. 1426–1438, 2006.

[24] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer Berlin Heidelberg, 2004.

[25] L. Parada, C. Herrera, M. Sepúlveda, and V. Parada, "Evolution of new algorithms for the binary knapsack problem," *Natural Computing*, vol. 15, no. 1, pp. 181–193, 2016.

[26] F. T. Leighton, *Introduction to parallel algorithms and architectures: Arrays· trees· hypercubes*. Elsevier, 2014.

[27] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[28] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1481–1484, 2017.

[29] J. Kim, J.-J. Lee, J.-K. Kim, and W. Lee, "Energy-efficient stabilized automatic control for multicore baseband in millimeter-wave systems," *IEEE Access*, vol. 5, pp. 16 584–16 591, 2017.

Nhu-Ngoc Dao received the B.S. degree in electronics and telecommunications from the Posts and Telecommunications Institute of Technology, Viet Nam, in 2009, and the M.S. degree in computer science from Chung-Ang University, South Korea, in 2016, where he is currently pursuing the Ph.D. degree in computer science. His research interests include network security, network softwarization, fog/edge computing, and Internet of Things.

Duc-Nghia Vu received his B.S. degree in Electronics and telecommunications from Hanoi University of Science and Technology, Viet Nam, in 2015. He also received the M.S. degree in computer science from Chung-Ang University, South Korea, in 2018. His research interests include wireless network and fog computing.

Woongsoo Na received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Chung-Ang University, Seoul, Korea, in 2010, 2012, and 2017, respectively. He is currently a adjunct professor in the School of information Technology at Sungshin University, Seoul, Korea. His research interests include mobile chargers, directional MAC, wireless mobile networks, and LTE.

Joongheon Kim received the B.S. and M.S. degrees in computer science and engineering from Korea University, Seoul, Korea, in 2004 and 2006, and the PhD degree in computer science from the University of Southern California (USC), Los Angeles, CA, in 2014, with two additional MS degrees in electrical engineering and computer science. He has been an assistant professor with Chung-Ang University, Seoul, Korea, since 2016. In industry, he was with LG Electronics (Seoul, Korea, 2006–2009), InterDigital (San Diego, CA, 2012), and Intel Corporation (Santa Clara, CA, 2013–2016). He is a senior member of the IEEE; and a member of the ACM and IEEE Communications Society. He was awarded the Annenberg graduate fellowship with his PhD admission from USC (2009).

Sungrae Cho is a professor with the school of software, Chung-Ang University (CAU), Seoul. Prior to joining CAU, he was an assistant professor with the department of computer sciences, Georgia Southern University, Statesboro, GA, USA, from 2003 to 2006, and a senior member of technical staff with the Samsung Advanced Institute of Technology (SAIT), Kiheung, South Korea, in 2003. From 1994 to 1996, he was a research staff member with electronics and telecommunications research institute (ETRI), Daejeon, South Korea. From 2012 to 2013, he held a visiting professorship with the national institute of standards and technology (NIST), Gaithersburg, MD, USA. He received the B.S. and M.S. degrees in electronics engineering from Korea University, Seoul, South Korea, in 1992 and 1994, respectively, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2002.

His current research interests include wireless networking, ubiquitous computing, and ICT convergence. He has been a subject editor of IET Electronics Letter since 2018, and an editor of Ad Hoc Networks Journal (Elsevier) from 2012 to 2017. He has served numerous international conferences as an organizing committee chair, such as IEEE SECON, ICOIN, ICTC, ICUFN, TridentCom, and the IEEE MASS, and as a program committee member, such as IEEE ICC, MobiApps, SENSORNETS, and WINSYS.