2019 IEEE International Conference on Web Services (ICWS)

A Mobility-Aware Cross-edge Computation Offloading Framework for Partitionable Applications

Hailiang Zhao*†, Shuiguang Deng*§, Cheng Zhang*, Wei Du†, Qiang He‡ and Jianwei Yin*
*College of Computer Science and Technology, Zhejiang Univeristy, Hangzhou, China
†School of Computer Science and Technology, Wuhan Univeristy of Technology, Wuhan, China
‡School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia
Email: hliangzhao97@gmail.com, {dengsg, coolzc, zjuyjw}@zju.edu.cn, whutduwei@whut.edu.cn, qhe@swin.edu.au

***Abstract*—Mobile Edge Computing has already become a new paradigm to reduce the latency in data transmission for resource-limited mobile devices by offloading computation tasks onto edge servers. However, for mobility-aware computation-intensive services, existing offloading strategies cannot handle the offloading procedure properly because of the lack of collaboration among edge servers. A data stream application is partitionable if it can be presented by a directed acyclic dataflow graph, which makes cross-edge collaboration possible. In this paper, we propose a cross-edge computation offloading (CCO) framework for partitionable applications. The transmission, execution and coordination cost, as well as the penalty for task failure, are considered. An online algorithm based on Lyapunov optimization is proposed to jointly determine edge site-selection and energy harvesting without priori knowledge. By stabilizing the battery energy level of each mobile device around a positive constant, the proposed algorithm can obtain asymptotic optimality. Theoretical analysis about the complexity and the effectiveness of the proposed framework is provided. Experimental results based on a real-life dataset corroborate that CCO can achieve superior performance compared with benchmarks where cross-edge collaboration is not allowed.**

***Keywords*-computation offloading, user mobility, cross-edge collaboration, application partitioning, energy harvesting.**

I. Introduction

Intelligent mobile devices (such as IoT sensors, wearable devices) have become indispensable tools in our daily routines. However, the limited computation and storage resources and battery capacities of mobile devices cannot meet the needs on various applications. In recent years, mobile edge computing (MEC) has emerged as a promising paradigm for overcoming these limitations. MEC provides various resources and services for mobile devices by pushing computing capabilities away from the centralized cloud to the network edge, in the vicinity of the widespread wireless access network [1].

In an MEC system, an edge site/server is a micro data center with applications depolyed, attached to a Small Base Station (SBS). User workloads can be offloaded from their mobile devices onto a nearby edge site for processing, which can reduce service cost measured by Quality of Experience (QoE). However, for scenarios where mobile devices can move randomly, the difficulty in designing the computation offloading strategy is high due to the frequent heterogeneous edge site selection and user profile handover.

Because the wireless signal coverages of SBSs often overlap in real-world scenarios, *a collaboration network* of edge sites can be constructed by the Mobile Network Operator (MNO). In addition, application partitioning and repartitioning have been in-depth studied in Mobile Cloud Computing and distributed systems [2], which can be put into use *selectively* in MEC systems. Following these two ideas, computation tasks can be processed cooperatively and distributively in a cross-edge way [3] by partitioning tasks and offloading them to multiple edge sites. However, what cannot be ignored is that the scheduling, communicating and coordinating costs increase. Therefore, it is critical to trade-off between the offloading & computing latency of mobile devices and the communication & coordination costs.

Apart from the communication and coordination costs, the offloading latency is also coupled with energy consumption of mobile devices because the performance may be compromised due to insufficient battery energy [4]. In many situations, it is not appropriate to recharge batteries frequently. For some outdoor sensors it is even impossible to recharge. Besides, frequent connections and data transmissions to more than one edge site in a time slot may lead to excessive *transient* discharge, which does great harm to the battery life of mobile devices [5]. Fortunately, a strong need for green computing prompts the development of Energy Harvesting (EH) technologies [6], which collect recyclable and clean energy, including wind, solar radiation, as well as human motion energy [7]. Therefore, it endows self-sustainability and perpetual operation of mobile devices.

In order to design an offloading strategy with the above concerns considered, this paper proposes a cross-edge computation offloading framework by forming a collaboration network of edge sites and processing *different parts* of a computation task on multiple servers simultaneously. This framework can achieve a near minimum latency with the battery energy of randomly moving mobile devices stabilizing at a reasonabe level. The main contributions of this

§Corresponding author.

978-1-7281-2717-0/19/$31.00 ©2019 IEEE
DOI 10.1109/ICWS.2019.00041

paper can be summarized as follows.

1) A cross-edge collaboration framework for computation offloading is proposed, which can be put into use for multi-user and multi-server MEC systems with *arbitrary partitionable applications*. The heterogeneity and computation resource limitation of edge sites, user mobility, overlapped signal coverage of SBSs, as well as the battery energy levels of mobile devices are all taken into consideration.

2) The trade-off between cross-edge collaboration cost and computation offloading latency is analyzed. Based on rigorous mathematical deducing, we verify that the proposed algorithms can achieve asymptotically optimal solutions by tuning the control parameters under the Lyapunov optimization technologies.

3) A simulation on a real-world dataset of base stations in the Melbourne CBD area [8] is conducted to verify the theoretical analysis. Moreover, the effectiveness of the proposed framework is demonstrated with a comparison with benchmark polices.

The organization of this paper is as follows. We survey the state of the art in Section II. In Section III, the system model is introduced. In Section IV, the cross-edge computation offloading framework is detailed and explained. The simulation results are demonstrated and analyzed in Section VI. Section VII concludes this paper.

II. Related Works

Most of researchers focus on the computation offloading problem in MCC systems and single-server MEC systems from different aspects such as QoS, QoE, resource (bandwidth, power, CPU-cycle frequency) management [9] [10] [11], etc.

Few in-depth studies, however, have devoted their efforts to the problem in *multi-user and multi-server* MEC systems [12] [13]. A representative research is [12], where resource competition and server selection were investigated for a device-edge system. This paper emphasises on how to greedily choose edge servers according to the legitimated bandwidth and computing resource allocation. However, signal coverage overlap and cross-edge collaboration are not considered.

User mobility management has been extensively studied in traditional heterogeneous cellular networks [14]. Deng et al. built service compositions in mobile communities when both service requesters and providers are mobile. Based on the proposed mobile service provisioning architecture, a service composition approach by utilizing the Krill-Herd algorithm is designed [15]. Furthermore, based on Lyapunov optimization, priori knowledge on mobility model is not necessary to know. A related work is [16], which decomposed a long-term optimization problem into a series of real-time problems without users' moving track.

There exists works dedicated on computation offloading with EH devices [17] [18] [19]. A deep Q-network-based strategic computation offloading algorithm was designed for a MDP in [18], where the objective was to minimize the long-term cost. However, the EH process only plays the role of connecting establishment between *states* in consecutive time slots. Mao et al. modeled the EH process as successive energy packet arrivals, which is adequate to capture the intermittent nature of vibrations and the renewable energy processes [19]. In our paper, the EH process is modeled as an i.i.d. uniform distribution.

Although [19] is the most similar work to our paper, serveral key differences should be addressed. Firstly, the problem addressed in [19] is much simpler, where the computation offloading was investigated in an MEC system with one single stationary mobile device and one single edge server. Secondly, user mobility and computation capability limitation of edge servers were ignored in [19]. Thirdly, in this paper we emphasis on the trade-off between cross-edge collaboration cost and offloading latency, which was not taken into consideration in [19]. Due to the differences mentioned above, our work is more complicated yet more realistic.

III. System Model

A. A Motivation Scenario

For discussion in this paper, we refer to an MEC system consisting of N mobile devices equipped with EH components, indexed by $\mathcal{N}$, and M SBSs, indexed by $\mathcal{M}$. SBSs are interconnected via X2 Link for data transmission and coordination [20]. The time horizon is discretized into time slots with length τ, indexed by $\mathcal{T} \triangleq \{1, 2, ...\}$.

Fig. 1 demonstrates an example. The system provides health monitoring and analytics for wearable devices, such as smart bracelets and intelligent glasses. Three wearable devices move in the manner of the Gauss-Markov mobility model across 6 time slots. At the beginning of each time slot, each wearable device generates a task offloading request with a certain probability, which can be successfully responsed to *iff* it does not timeout, otherwise the request will be dropped. Firstly, the preprocessing and packing of monitored data are carried out by wearable devices. Next, the preprocessed data are divided equally and offloaded to chosen edge sites for cross-edge analytics. In each time slot, harvestable energy comes from light, kinetic, wind, and other vibrations. The EH components are implemented in the same manner as in [21]. Without aggregating geo-distributed data to a centralized data center, health analytics are carried out by edge sites via cross-edge Map-Reduce queries [22]. Considering that we focus on edge site selection, *it is assumed that* edge sites can collaborate with each other in perfect match, i.e., transitions of each application phase are instantaneous without failure. Relative to the offloaded data, the returned analytical results are much smaller. Therefore, the latency in the downlink transmission is ignored.

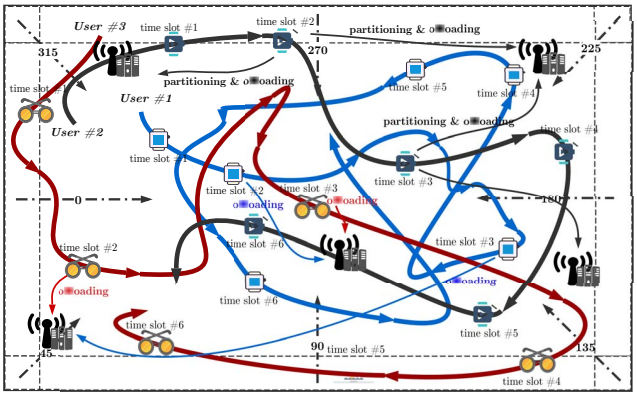


Figure 1: A motivation scenario.

B. Local Execution Latency Evaluation

For the entire time horizon, each SBS's location is fixed while each mobile device can move around in *any number of ways* in different time slots. Let us denote the set of SBSs covering the ith mobile deivce in time slot t as $\mathcal{M}_i(t)$. Correspondingly, $\mathcal{N}_j(t)$ denotes the set of mobile devices covered by edge site j in the tth time slot.

We model the task offloading demands of mobile devices as an i.i.d. Bernoulli distribution. In each time slot, the ith mobile device's offloading demand is generated with probability ρ_i. We set $\mathbf{A}(t) \triangleq \{\times_{i\in\mathcal{N}} A_i(t)\} \subseteq \{0,1\}^N$ as the demand vector, i.e., $\Pr\{A_i(t) = 1\} = \rho_i$. The local execution includes data preprocessing and packing. We set the data size to be processed for local execution and offloadingas μ_i^l and μ_i^r, respectively. Correspondingly, the two parts need η_i^l and η_i^r CPU cycles, respectively. For local execution, the execution latency τ_i^{lc} is η_i^l/f_i. The energy consumption of local execution is

$$\epsilon_i^l = \kappa_i \cdot \eta_i^l f_i^2, i \in \mathcal{N}, t \in \mathcal{T}, \tag{1}$$

where κ_i is the effective switched capacitance that depends on the chip architecture.

C. Offloading Latency Evaluation

Let us denote $\mathbf{I}_i(t) \triangleq [I_{i,1}(t), ..., I_{i,|\mathcal{M}_i(t)|}(t)], I_{i,j}(t) \in \{0,1\}$ as the edge site-selection indicator. We assume that the jth edge site can be assigned to at most N_j^{max} mobile devices due to its limited computational capability, which means we have the following constraint:

$$\sum_{i\in\mathcal{N}} I_{i,j}(t) \leq N_j^{max}, j \in \mathcal{M}_i(t), t \in \mathcal{T}. \tag{2}$$

For each mobile device i, the latency in offloading comes from the wireless uplink transmission, computation and edge sites collaboration. Denote the small-scale fading channel power gains from the ith mobile device to the jth SBS by $\zeta_{i,j}(t)$. The channel power gain can be obtained by $h_{i,j}(t) \triangleq \zeta_{i,j}(t) g_0 (\frac{d_0}{d_{i,j}(t)})^\lambda$, where d_0 and $d_{i,j}(t)$ denote the reference distance and real distance between i and j, respectively. λ denotes the pass-loss exponent and g_0 denotes the pass-loss constant. As a result, the achievable rate of from the ith mobile device to the jth edge site $R_{i,j}(t)$ is

$$R_{i,j}(t) \triangleq \omega \log_2 \left(1 + \frac{h_{i,j}(t) p_i^{tx}}{I + \varpi_0}\right), j \in \mathcal{M}_i(t), \tag{3}$$

where ω represents the bandwidth allocated, I is the maximum received average power of interference and ϖ_0 is the addictive background noise. p_i^{tx} represents the fixed transmit power. Therefore, the transmission latency can be obtained by

$$\tau_{i,j}^{tx}(t) = \frac{\mu_i^r}{\sum_{j\in\mathcal{M}_i(t)} I_{i,j}(t)} \cdot \frac{1}{R_{i,j}(t)}, j \in \mathcal{M}_i(t). \tag{4}$$

More complicated data partitioning models will be studied in future. Based on (3) and (4), the energy consumption of transmission $\epsilon_{i,j}^{tx}(t)$ is $p_i^{tx} \cdot \tau_{i,j}^{tx}(t)$. Similarly, the latency of computing $\tau_{i,j}^{rc}(t)$ is

$$\tau_{i,j}^{rc}(t) = \frac{\eta_i^r}{f_j \cdot \sum_{j\in\mathcal{M}_i(t)} I_{i,j}(t)}, j \in \mathcal{M}_i(t), \tag{5}$$

where f_j is the CPU cycle frequency of edge site j. Without loss of generality, $\forall i \in \mathcal{N}, j \in \mathcal{M}$, we set $f_j \gg f_i$. In order to execute the computation task successfully, we need to follow the constraint below:

$$\begin{aligned} \tau_d \quad \geq \quad & \max_{j\in\mathcal{M}_i(t)} \{\tau_{i,j}^{tx}(t) + \tau_{i,j}^{rc}(t)\} \\ + \quad & \tau_i^{lc} + \varphi \cdot \sum_{j\in\mathcal{M}_i(t)} I_{i,j}(t), \end{aligned} \tag{6}$$

where τ_d is the execution deadline. Without loss of generality, we set $\tau_d \leq \tau$. Although the transmission latency can be reduced if multiple edge sites are chosen, the coordination cost is propotional to the number of chosen edge sites. We simply assume that the coordination cost is propotional to the number of chosen edge sites. Thus it can be described as $\varphi \cdot \sum_{j\in\mathcal{M}_i(t)} I_{i,j}(t)$, where φ is the unit latency cost. Apperantly, it is critical to trade-off between the cross-edge collaboration cost and the offloading latency.

D. Battery Energy Level Evaluation

Denote the battery energy level of the ith mobile device at the beginning of the tth time slot as $\psi_i(t)$. We set $\psi_i(T) < +\infty$ as $T \to +\infty, i \in \mathcal{N}$. Actually, finite battery energy capacity is also feasible, which will be detailed in [23]. The energy consumption has the following constraint:

$$\epsilon_i^l + \sum_{j\in\mathcal{M}_i(t)} \epsilon_{i,j}^{tx}(t) I_{i,j}(t) \leq \psi_i(t), i \in \mathcal{N}, t \in \mathcal{T}. \tag{7}$$

Successive energy packets with size $E_i^h(t)$ arrival at the beginning of each time slot. We assume $E_i^h(t)$ is i.i.d. with the maximum value of $E_{i,h}^{max}$ in different time slots. Denote $\alpha_i(t)$ as the amount of energy unit to be stored in mobile device. Thus the evolution equation of energy for the ith

mobile device is

$$\psi_i(t+1) = \psi_i(t) - \sum_{j \in \mathcal{M}_i(t)} \epsilon_{i,j}^{tx}(t) \cdot I_{i,j}(t) - \epsilon_i^l + \alpha_i(t), \quad (8)$$

where

$$0 \leq \alpha_i(t) \leq E_i^h(t), i \in \mathcal{N}, t \in \mathcal{T}. \quad (9)$$

IV. Problem Formulation

A. Edge Site-selection Problem with Parallel Execution

The total energy consumption of the ith mobile device is $\epsilon_i^l + \sum_{j \in \mathcal{M}} \epsilon_{i,j}^{tx}(t) \cdot I_{i,j}(t)$. When it excessively discharges, the battery life is affected. More importantly, there may be safety problems. Therefore, we introduce *safe discharge threshold*, i.e., the maximum transient discharge w.r.t. the ith mobile device, as ψ_i^{safe}. Thus, we have

$$\epsilon_i^l + \sum_{j \in \mathcal{M}_i(t)} \epsilon_{i,j}^{tx}(t) I_{i,j}(t) \leq \psi_i^{safe}, i \in \mathcal{N}, t \in \mathcal{T}. \quad (10)$$

Besides, it is possible that there has no feasible solution with *insufficient* battery energy level under (6). As we have described in III-A, those timeout requests have to be dropped. We set $D_i(t) \in \{0, 1\}$ as the indicator of dropping the task, i.e., $D_i(t) = \mathbb{1}\{\mathbf{I}_i(t) = \mathbf{0}\}$. When the request of the ith mobile device in the tth time slot is dropped, a penalty ϱ_i is generated and applied. We set $\varrho_i \geq \tau_d$, which means that a task is preferred to be executed successfully than be dropped.

Denote $\boldsymbol{\epsilon}_i(t)$ as the offloading energy allocation vector with size $|\{j \in \mathcal{M} | \mathbb{1}\{I_{i,j} = 1\}\}|$. The overall cost of the ith mobile device in the tth time slot is

$$\begin{aligned} \mathcal{C}(\mathbf{I}_i(t)) \quad &\triangleq \quad \max_{j \in \mathcal{M}_i(t): I_{i,j'}(t)=1} \left\{ \tau_{i,j}^{tx}(t) + \tau_{i,j}^{rc}(t) \right\} \\ &+ \quad \tau_i^{lc} + \varphi \cdot \sum_{j \in \mathcal{M}_i(t)} I_{i,j}(t) + \varrho_i \cdot D_i(t). \end{aligned} \quad (11)$$

Notice that this is a general model without specific structural assumptions. In practice, a proper value of φ can be determinated according to the types of coordination through *Multiple Criteria Decision Making*. Consequently, the *Edge Site-selection Problem* can be formulated as follows:

$$\begin{aligned} \mathcal{P}_1 : \quad \min_{\forall i, \mathbf{I}_i(t), \alpha_i(t)} \quad & \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\sum_{i \in \mathcal{N}} \mathcal{C}(\mathbf{I}_i(t)) \right] \\ s.t. \quad & (2), (6), (7), (9), (10). \end{aligned}$$

B. Problem Analysis

In the considered problem, the system state is composed of the task request, the harvestable energy, as well as the battery energy level, and the action lies in edge site-selection and energy harvesting. It can be checked that the allowable action set depends only on the current system state, and is irrelevant with the state and the action history. Therefore, $\mathcal{P}_1$ is a MDP. In principle, it can be solved optimally with standard MDP algorithms. However, the system first needs to be discretized, and obviously the feasible solution space is too huge to be traversed in a rational time. In addition, what cannot be ignored is that the huge memory requirement for storing the optimal policy with the discretized system. Deep Q-Network, as a model-free learning algorithm, can be utilized to solve the problem directly. However, quantizing the *state* and the *action* may lead to severe performance degradation.

In this paper, we propose several centralized algorithms without priori knowledge on random events happening in this system. Those approaches are based on Lyapunov Optimization and the SAC algorithm [24]. Theoretically, by adjusting parameters V and θ_i (defined in Section V), the proposed algorithms can achieve solutions arbitrarily close to the optimal solution to $\mathcal{P}_1$ with finite battery energy capacity. Specifically, for a highly non-convex sub-problem, we propose a SAC-based algorithm to obtain the near optimal solution, which can achieve a polynomial improvement over the uniform search on the discretized solution space. Details on theoretical analysis can be consulted in online appendix [23].

V. The Cross-edge Computation Offloading Framework

A. Framework Overview

Fig. 2 demonstrates the architecture of the cross-edge computation offloading framework. The centralized framework consists of mobile devices, edge sites (edge server + SBS) and the Mobile Network Operator (MNO). The resource manager is the core component for CCO algorithm implementation, where CCO algorithm is the main algorithm executed across the time horizon. Details can be found in Subsection V-B. In each time slot, the mobile devices generate the computation task requests with a certain probability distribution. Those requests along with the basic information of senders (e.g., app type, local CPU-cycle frequency, battery energy level, etc.) will be sent to the MNO. After that, the MNO monitors the channel state information of edge sites for dynamic scaling. By executing CCO algorithm, the MNO chooses edge sites for each mobile device for offloading. Finally, edge sites output the analytical results to the MNO to determine the final downlink transmission.

B. Cross-edge Computation Offloading Algorithm

In this subsection, we demonstrate the details on how to obtain the asymptotic optimal solution of $\mathcal{P}_1$ by our CCO algorithm.

We use a vector $\boldsymbol{\Theta}(t) \triangleq [\psi_1(t), ..., \psi_N(t)]$ to represent the system energy queues in the tth time slot. For a given set of non-negative parameters $\boldsymbol{\theta} \triangleq [\theta_1, ..., \theta_N]$, the non-negative *Lyapunov function* $L(\boldsymbol{\Theta}(t))$ is defined as follows:

$$L(\boldsymbol{\Theta}(t)) \triangleq \frac{1}{2} \sum_{i=1}^{N} (\psi_i(t) - \theta_i)^2 = \sum_{i=1}^{N} \psi_i'(t)^2, \quad (12)$$

where

$$\theta_i \geq \frac{V\cdot\omega\log_2(1+\frac{h_{i,j}^{max}p_i^{tx}}{\varpi_0})}{p_i^{tx}\mu_i^r}\Bigg(-\frac{\mu_i^r}{\omega\log_2(1+\frac{h_{i,j}^{max}p_i^{tx}}{\varpi_0})} + M\varrho_i-\frac{\eta_i^r}{f_j}\Bigg)+\min\{E_{i,all}^{max},\psi_i^{safe}\}$$

and $E_{i,all}^{max}\triangleq\epsilon_i^l+\sum_{j=1}^{M}\epsilon_{i,j}^{max}, \epsilon_{i,j}^{max}\triangleq p_i^{tx}(\tau_d-\tau_i^{lc}), h_{i,j}^{max}\triangleq \max_{t\in\mathcal{T}:j\in\mathcal{M}} h_{i,j}(t)$. (12) tends to keep battery energy backlog near a non-zero value θ_i for the ith mobile device. Define the conditional Lyapunov drift $\Delta(\mathbf{\Theta}(t))$ as

$$\Delta(\mathbf{\Theta}(t))\triangleq\mathbb{E}[L(\mathbf{\Theta}(t+1))-L(\mathbf{\Theta}(t))|\mathbf{\Theta}(t)]. \quad (13)$$

Notice that the lower bound of θ_i presented above is not a tightened version. Actually, the larger battery capacity of mobile devices, the more optimized solution can be obtained. Due to the limited length of the paper, in-depth analysis is provided in the online appendix: [23].

According to (8), we can obtain that

$$\psi_i'(t+1)^2 \leq \psi_i'(t)^2+2\psi_i'(t)\big[\alpha_i(t)-\epsilon_i^l-\textstyle\sum_{j\in\mathcal{M}}\epsilon_{i,j}^{tx}(t)\cdot I_{i,j}(t)\big] +(E_{i,h}^{max})^2+(E_{i,all}^{max})^2.$$

Thus we have

$$\Delta(\mathbf{\Theta}(t))\leq\sum_{i=1}^{N}\psi_i'(t)\big[\alpha_i(t)-\epsilon_i^l-\sum_{j\in\mathcal{M}}\epsilon_{i,j}^{tx}(t)\cdot I_{i,j}(t)\big]+C,$$

where $C\triangleq\frac{1}{2}\sum_{i=1}^{N}\big[(E_{i,h}^{max})^2+(E_{i,all}^{max})^2\big]$. By Lyapunov optimization, we can obtain the near optimal solution to $\mathcal{P}_1$ by minimizing the upper bound of $\Delta(\mathbf{\Theta}(t))+V\cdot\sum_{i=1}^{N}\mathcal{C}(\mathbf{I}_i(t))$, without regard to (7). With $\Delta_V^{up}(\mathbf{\Theta}(t))$ defined by

$$\Delta_V^{up}(\mathbf{\Theta}(t)) \triangleq \sum_{i=1}^{N}\psi_i'(t)\big[\alpha_i(t)-\epsilon_i^l-\sum_{j=1}^{M}\epsilon_{i,j}^{tx}(t)I_{i,j}(t)\big] + V\sum_{i=1}^{N}\mathcal{C}(\mathbf{I}_i(t))+C, \quad (14)$$

we introduce the deterministic problem $\mathcal{P}_2$ in the every time slot t as

$$\mathcal{P}_2: \min_{\forall i,\mathbf{I}_i(t),\alpha_i(t)} \Delta_V^{up}(\mathbf{\Theta}(t)) \quad s.t. \quad (2),(6),(9),(10),$$

where the constraint (7) is ignored because it violates the conditions of vanilla version of Lyapunov optimization for i.i.d. random events [25]. To simplify the problem, we set $\psi_i^{safe}\geq E_{i,all}^{max}$, which means (10) can be reasonably ignored. So far, we can demonstrate the details of our CCO Algorithm, as shown in **Algorithm 1**.

The most important part of CCO algorithm lies in how to solve $\mathcal{P}_2$ asymptotic optimally. Actually, $\mathcal{P}_2$ can be divided into two sub-problems: *optimal energy harvesting* and *optimal edge site-selection*.

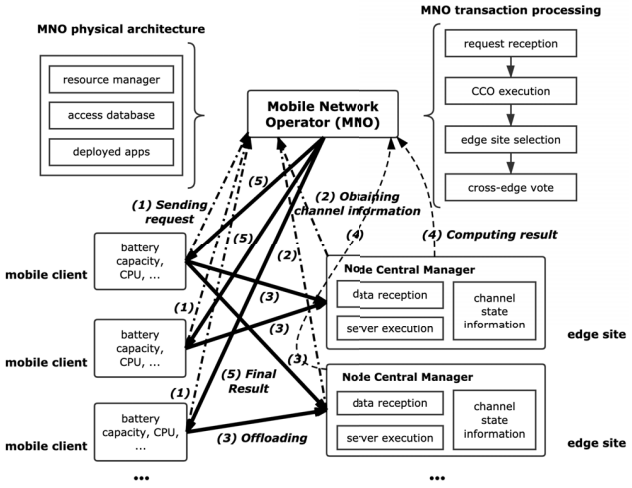


Figure 2: The cross-edge computation offloading framework.

Optimal energy harvesting. The optimal amount of harvested energy $\alpha_i^\star(t)$ can be obtained by solving the following sub-problem:

$$\mathcal{P}_2^{EH}: \min_{\forall i,\alpha_i(t)} \sum_{i=1}^{N}\psi_i'(t)\alpha_i(t) \quad s.t. \quad (9).$$

It is easy to obtain that

$$\alpha_i^\star(t)=E_i^H(t)\cdot\mathbb{1}\{\psi_i'(t)\leq 0\}, i\in\mathcal{N}. \quad (15)$$

Optimal edge site-selection. The optimal decision vector $\mathbf{I}_i^\star(t)$ can be obtained by solving the following sub-problem:

$$\mathcal{P}_2^{es}: \min_{\forall i,\mathbf{I}_i(t)} \Bigg\{-\sum_{i=1}^{N}\psi_i'(t)\Bigg[\epsilon_i^l+\sum_{j\in\mathcal{M}_i(t)}\epsilon_{i,j}(t)I_{i,j}(t)\Bigg] + V\cdot\sum_{i=1}^{N}\mathcal{C}(\mathbf{I}_i(t))\Bigg\} \quad s.t. \quad (2),(6),(10).$$

Algorithm 1 Cross-edge Computation Offloading (CCO)

1: At the beginning of the tth time slot, obtain i.i.d. random events $\mathbf{A}(t),\mathbf{E}^h(t)\triangleq[E_1^h(t),...,E_N^h(t)]$ and channel state information.
2: $\forall i\in\mathcal{N}$, decide $\mathbf{I}_i^\star(t),\alpha_i^\star(t)$ by solving the deterministic problem $\mathcal{P}_2$.
3: $\forall i\in\mathcal{N}$, update the battery energy level $\psi_i(t)$ by (8).
4: $t\leftarrow t+1$.

As a highly non-convex combinatorial optimization problem, it is difficult to obtain the optimal solution of $\mathcal{P}_2^{es}(t)$ by the *branch and bound method* with commercial software

directly. Inspired by *the Sampling-and-Classification (SAC) algorithm* and its derived algorithm, e.g., RACOS [24], we propose the following SAC-based Edge site-Selection (SES) algorithm, as shown in **Algorithm 2**. Under the *error-target independence* proposed in **Definition 2** of [24], which assumes that the error of the learned classifier/hypothesis h in each iteration is independent with the target approximation area, the SES algorithm can obtain a *polynomial* improvement over the uniform search on the *positive solution set* $\mathcal{D}_h \triangleq \{\mathbf{I}(t)|h(\mathbf{I}(t)) = +1\}$. A tightened query complexity of SAC algorithms in discrete domains is presented in **Theorem 1** of [24].

The SES algorithm first initializes the feasible solution set $\boldsymbol{\mathcal{S}}_0(t) = \{\mathbf{I}^1(t), ..., \mathbf{I}^s(t), ..., \mathbf{I}^{S_0}(t)\}$ by i.i.d. sampling from the uniform distribution over solution space $\{0,1\}^{\times_{i\in\mathcal{N}}\mathcal{M}_i(t)}$ with constraint (2) embedded (line 1 to 5), where $\mathbf{I}^s(t) \triangleq \{\times_{i\in\mathcal{N}}\mathbf{I}_i(t)\}$ is the sth generated solution. The way of sampling from the feasible solution set follows the idea that we maximize the utilization of computation and communication resources of each edge site. For the mobile device who misses the computation task' deadline or whose battery energy is insufficient, the task has to be dropped, i.e., $D_i(t) \leftarrow 1$. After that, a cycle is followed (line 9 to 18). In each iteration, SES algorithm queries the objective function $G_{\mathcal{P}_2^{es}}$ to assess the geneated solutions, and then forms a binary classification dataset $\mathcal{Q}_k(t)$, where a threshold γ_k is used to label the solution as $+1$ and -1. In the classification phase (line 11), a binary classifier is trained on $\mathcal{Q}_k(t)$, in order to approximate the region $\mathcal{D}_{\gamma_k} \triangleq \{\mathbf{I}(t) \in \boldsymbol{\mathcal{S}}_k(t)|G_{\mathcal{P}_2^{es}} \leq \gamma_k\}$. During the sampling phase (line 14 to 15), solutions are sampled from distribution $\mathcal{H}_{h_k}$ and the universal set of feasible solutions tuned by ε. As [26] has pointed out, uniform sampling with the positive region $\mathcal{D}_{h_k}$ is straightforward and efficient, i.e., $\mathcal{H}_{h_k} \leftarrow \mathcal{U}_{h_k}$. Throughout the procedure, the best-so-far solutions are recorded (line 6 and 17), and the best solution will be returned as the output (line 19).

Algorithm 2 SAL-based Edge site-Selection (SES)

1: **for** $s = 1$ to S_0 **do**
2: *Sample the sth solution* $\mathbf{I}_s(t)$ *from the feasible solution space* X *(denoted as* $\mathcal{U}_X$*):* $\forall j \in \mathcal{M}$, assign $\min\{N_j^{max}, |\mathcal{N}_j(t)|\}$ connections to elements in set $\mathcal{N}_j(t)$ randomly.
3: $\forall i \in \mathcal{N}$, update $D_i(t)$ as $\mathbb{1}\{\big(\max_{j\in\mathcal{M}_i(t)}\{\tau_{i,j}^{tx}(t) + \tau_{i,j}^{rc}(t)\} + \tau_i^{lc} + \varphi\cdot\sum_{j\in\mathcal{M}_i(t)} I_{i,j}(t) > \tau_d\big) \vee \big(\boldsymbol{\epsilon}_i(t) < \psi_i(t)\big)\}$.
4: For those mobile devices who satisfy $D_i(t) = 1$, update $\mathbf{I}_i(t)$ as $\mathbf{0}$.
5: **end for**
6: $\mathbf{I}^\star(t) \leftarrow \mathrm{argmin}_{\mathbf{I}^s(t)\in\boldsymbol{\mathcal{S}}_0(t)} G_{\mathcal{P}_2^{es}}(\mathbf{I}^s(t))$,
7: Initialize the hypothesis h_0.
8: $\mathcal{Q}_0(t) \leftarrow \varnothing$.
9: **for** $k = 1$ to K **do**
10: *Construct the binary-labeled dataset:* For all $\mathbf{I}^s(t) \in \boldsymbol{\mathcal{S}}_{k-1}(t), y^s(t) = \mathrm{sign}\{\gamma_k - G_{\mathcal{P}_2^{es}}(\mathbf{I}^s(t))\}$, $\mathcal{Q}_k(t) \triangleq \{(\mathbf{I}^1(t), y^1(t)), ..., (\mathbf{I}^{S_{k-1}}(t), y^{S_{k-1}}(t))\}$ where γ_k is the threshold for labeling.
11: Obtain the hypothesis with a binary classification algorithm $\mathcal{L}(\cdot)$: $h_k \leftarrow \mathcal{L}(\mathcal{Q}_k(t))$.
12: Initialize $\boldsymbol{\mathcal{S}}_k(t)$ as $\varnothing$.
13: **for** $s = 1$ to S_k **do**
14: *Sample with* ε*-greedy policy:*

$$\text{Get } \mathbf{I}^s(t) \text{ from } \begin{cases} \mathcal{H}_{h_k}, & \text{with probability } \varepsilon \\ \mathcal{U}_X, & \text{with probability } 1-\varepsilon, \end{cases}$$

where $\mathcal{H}_{h_k}$ is the distribution transformation of hypothesis h_k.
15: $\boldsymbol{\mathcal{S}}_k(t) \leftarrow \boldsymbol{\mathcal{S}}_k(t) \cup \{\mathbf{I}_s(t)\}$.
16: **end for**
17: $\mathbf{I}^\star(t) \leftarrow \mathrm{argmin}_{\mathbf{I}^s(t)\in\boldsymbol{\mathcal{S}}_k(t)\cup\{\mathbf{I}^\star(t)\}} G_{\mathcal{P}_2^{es}}(\mathbf{I}^s(t))$.
18: **end for**
19: **return** $\mathbf{I}^\star(t)$.

VI. Experimental Evaluation

This section evaluates the performance of our CCO algorithm and SES algorithm against three baselines.

A. Benchmark policies

Due to the fact the CCO algorithm is the first attempt on the *cross-edge* computation offloading problem in multi-user and multi-server scenarios, existing algorithms cannot be applied directly. Therefore, three intuitive baseline algorithms, namely Random Selection, Greedy Selection on Communication, and Greedy Selection on Computation are designed to be compared with our CCO algorithm.

1) **Random Selection (RS):** In every time slot, this algorithm randomly chooses edge sites to offload.

2) **Greedy Selection on Communication (GSC1):** In every time slot, each mobile device offloads computation tasks to the nearest edge site. Cross-edge offloading is not permitted. If the chosen edge site has been allocated for another mobile device, the edge site with the second maximum channel power gain will be selected. The process ends when all $\mathbf{I}_i(t)$ are set.

3) **Greedy Selection on Computation (GSC2):** In every time slot, each edge site distributes all of its computation capacity to its signal-covered mobile devices.

B. Experimental Settings

In the experiments, computation offloading scenarios are simulated based on the geolocation information about 126 edge sites within the Melbourne CBD area contained in the EUA dataset [8]. The coverage radius of each SBS is sampled following the uniform distribution $[150, 400]$ meters. System parameters are setted after in-depth investigation, as presented in Table. I. η_i^l and η_i^r are decided based on the

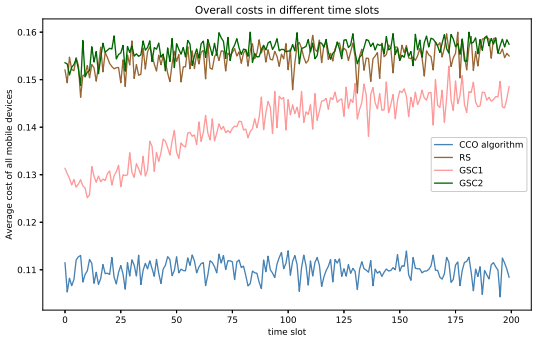


Figure 3: Average cost of mobile devices.

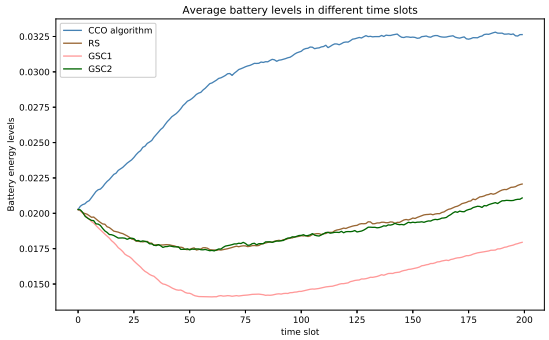


Figure 4: Average battery energy level of mobile devices.

fact that 737.5 cycles are needed for executing one bit of task. By defualt, we set the number of mobile devices to 80 and their initial batery energy level to $\frac{\theta_i}{2}$. User mobility is implemented based on random walk. Besides, the overall cost $\sum_{i\in\mathcal{N}}\mathcal{C}(\mathbf{I}_i(t))$ is empolyed as the performance for the evaluation.

Table I: System parameters.

Parameter	Value	Parameter	Value
τ_d	2 ms	ϱ_i	2 ms
φ	0.02 ms	ρ_i	0.6
μ_i^l	100 bits	μ_i^r	3000 bits
f_i	1.5 GHz	f_j	32 GHz
κ_i	10^{-28}	ψ_i^{safe}	40 mJ
N_j^{max}	5	ω	$1.5/\sum_{i\in\mathcal{N}_j(t)} I_{i,j}(t)$ GHz
ϖ_0	10^{-13} W	p_i^{tx}	1 W
g_0	10^{-4}	$E_{i,h}^{max}$	4.8×10^{-4} J

C. Experiment Results

Optimality and stability. As shown in Fig. 3, the CCO algorithm outperforms the three baseline algorithms significantly in overall cost $\sum_{i\in\mathcal{N}}\mathcal{C}(\mathbf{I}_i(t))$. Specifically, it outperforms RS, GSC2, and GCS1 by 36.40-42.92%, 17.83-36.96%, and 37.76-45.19%, respectively. In terms of GCS2, compared with the saved computation latency through cross-edge computation offloading, th communication speed is degraded becasue the allocated bandwidth is shared by more mobile devices. Therefore, GSC2 cannot outperform RS. The performance of GSC1 is worse than our CCO algorithm but better than RS and GSC2, which indicates that communicaiton has a greater influence on the overall latency. As shown in Fig. 4, the CCO algorithm can stabilize the battery energy level of each mobile device around θ_i and hold $\psi_i(t)\in[0,\theta_i+E_{i,h}^{max}]$. This result verifies **Lemma 1** in [23]. Apperantly, this stability is not possessed of by RS, GSC1, and GSC2.

Impacts of important parameters.

1) Impacts of control parameter V. As shown in Fig. 5(a), the overall cost achieved by the CCO algorithm decreases with the increase in V. But the performance of RS, GSC1, and GSC2 are not affected. Our CCO algorithm always outperforms the baseline algorithms. Inversely, as shown in Fig. 5(c), the battery energy level of each mobile device increases as V increase. This indicate that the solution obtained by the CCO algorithm is $O(\frac{1}{V})$-derivated from the optimality and the system queue size is $O(V)$, which verifies **Theorem** 1 in [23].

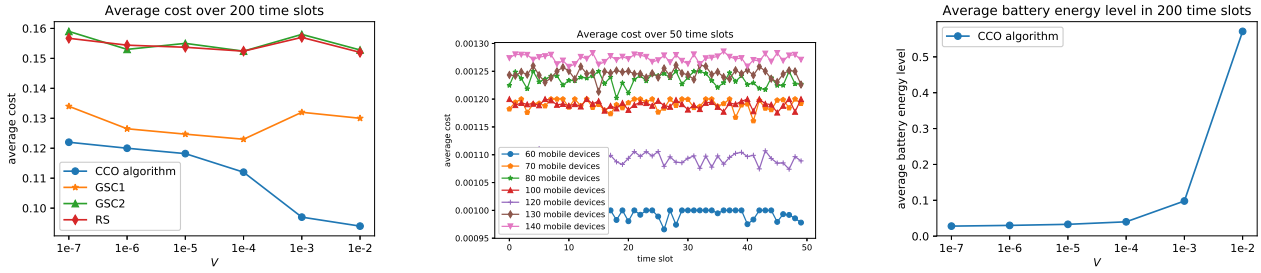
2) Impacts of the number of mobile devices N. The data fluctuation over different N is interesting. As shown in Fig. 5(b), when N increases from 60 to 80, the average cost increases. But when N increases from 80 to 120, the average cost decreases. When N increases from 120 to 140, the average cost increases again. The overall costs obtained by CCO algorithm is not monotonic with N. This happens because of the trade-off between more intense competition on bandwidth and more efficient utilization on computing resources among mobile devices.

VII. Conclusion

In this paper, We designed a centralized cross-edge computation offloading framework for partitionable applicaitons with edge site coordination cost considered. The framework can dynamically scale edge site-selection by leveraging various elastic resources. We proposed an algorithm based on the Lyapunov optimization techniques, which can obtain asymptotical optimality with battery capacity of mobile devices stabilizing around a positive constant. Specifically, the SES algorithm is proposed with a polynomial improvement over the uniform search on the feasible solution space. The experimental results based on real-world dataset show that our algorithm outperforms three baseline algorithms. In future, we will study typical real-world applicaiton scenarios. Besides, how to design a *distributed* cross-edge framework that lifts the heavy computation burden of MNO will be our research emphasis.

Acknowledgement

This research was partially supported by the National Key Research and Development Program of China (No. 2017YFB1400601), Key Research and Development Project



(a) Overall costs of all mobile devices vs. V.

(b) Average cost of all mobile devices vs. N over 50 time slots.

(c) Average battery energy level of all mobile devices vs. V.

Figure 5: Impacts of important parameters.

of Zhejiang Province (No. 2017C01015), National Science Foundation of China (No. 61772461), Natural Science Foundation of Zhejiang Province (No. LR18F020003 and No.LY17F020014), the Ministry of Education Project of Humanities and Social Sciences (No. 16YJCZH014), and Australian Research Council Discovery Projects (DP 170101932 and DP 18010021).

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017.

[2] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253–2266, Aug 2015.

[3] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven iot service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54 258–54 269, 2018.

[4] S. Deng, H. Wu, W. Tan, Z. Xiang, and Z. Wu, "Mobile service selection for composition: An energy consumption perspective," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 3, pp. 1478–1490, July 2017.

[5] T. Kim, W. Qiao, and L. Qu, "A series-connected self-reconfigurable multicell battery capable of safe and effective charging/discharging and balancing operations," in *2012 27th Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, Feb 2012, pp. 2259–2264.

[6] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, P. Grover, and K. Huang, "Energy harvesting wireless communications: A review of recent advances," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 3, pp. 360–381, March 2015.

[7] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Communications Surveys Tutorials*, vol. 13, no. 3, pp. 443–461, Third 2011.

[8] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Service-Oriented Computing*, C. Pahl, M. Vukovic, J. Yin, and Q. Yu, Eds. Cham: Springer International Publishing, 2018, pp. 230–245.

[9] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[10] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. PP, pp. 1–1, 10 2018.

[11] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3317–3329, Dec 2015.

[12] H. Zhao, W. Du, W. Liu, T. Lei, and Q. Lei, "Qoe aware and cell capacity enhanced computation offloading for multi-server mobile edge computing systems with energy harvesting devices," in *2018 IEEE International Conference on Ubiquitous Intelligence Computing*, Oct 2018, pp. 671–678.

[13] C. Zhang, H. Zhao, and S. Deng, "A density-based offloading strategy for iot devices in edge computing systems," *IEEE Access*, vol. 6, pp. 73 520–73 530, 2018.

[14] D. Lopez-Perez, I. Guvenc, and X. Chu, "Mobility management challenges in 3gpp heterogeneous networks," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 70–78, December 2012.

[15] S. Deng, L. Huang, J. Taheri, J. Yin, M. Zhou, and A. Y. Zomaya, "Mobility-aware service composition in mobile communities," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 3, pp. 555–568, March 2017.

[16] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, Oct 2018.

[17] M. Min, D. Xu, L. Xiao, Y. Tang, and D. Wu, "Learning-based computation offloading for iot devices with energy harvesting," *CoRR*, vol. abs/1712.08768, 2017.

[18] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," *CoRR*, vol. abs/1804.00514, 2018.

[19] 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, Dec 2016.

[20] C. B. Networks, "Backhauling x2," http://cbnl.com/resources/backhauling-x2/, accessed Feb 2, 2018.

[21] M. Magno and D. Boyle, "Wearable energy harvesting: From body to battery," in *2017 12th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS)*, April 2017, pp. 1–6.

[22] L. Jia, Z. Zhou, and H. Jin, "Optimizing the performance-cost tradeoff in cross-edge analytics," in *2018 IEEE International Conference on Ubiquitous Intelligence Computing*, October 2018, pp. 564–571.

[23] "Theoretical analysis for cross-edge computation offloading," https://narcissushliangzhao.github.io/Curriculum-Vitae/papers/Theoretical_analysis.pdf, accessed April 19, 2019.

[24] Y. Yu, H. Qian, and Y.-Q. Hu, "Derivative-free optimization via classification," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, pp. 2286–2292.

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

[26] H. Qian and Y. Yu, "On sampling-and-classification optimization in discrete domains," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 4374–4381.