Received: 27 July 2020 | Revised: 12 October 2020 | Accepted: 10 November 2020

DOI: 10.1002/spe.2942

SPECIAL ISSUE PAPER

WILEY

Exploring reliable edge-cloud computing for service latency optimization in sustainable cyber-physical systems

Kun Cao[1] | **Tongquan Wei**[2] | **Mingsong Chen**[3] | **Keqin Li**[4] | **Jian Weng**[1] | **Wuzheng Tan**[1]

[1]College of Information Science and Technology/College of Cyber Security, Jinan University, Guangzhou, China

[2]School of Computer Science and Technology, East China Normal University, Shanghai, China

[3]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China

[4]Department of Computer Science, State University of New York, New Paltz, New York, USA

Correspondence
Wuzheng Tan, College of Information Science and Technology/College of Cyber Security, Jinan University, Guangzhou 510632, China.
Email: tanwuzheng@126.com

Abstract

In recent years, the advance in information technology has promoted a wide span of emerging cyber-physical systems (CPS) applications such as autonomous automobile systems, healthcare monitoring, and process control systems. For these CPS applications, service latency management is extraordinarily important for the sake of providing high quality-of-experience to terminal users. Edge-cloud computing, integrating both edge computing and cloud computing, is regarded as a promising computation paradigm to achieve low service latency for terminal users in CPS. However, existing latency-aware edge-cloud computing methods dedicated for CPS fail to jointly consider energy budgets and reliability requirements, which may greatly degrade the sustainability of CPS applications. In this article, we explore the problem of minimizing service latency of edge-cloud computing coupled CPS under the constraints of energy budgets and reliability requirements. We propose a two-stage approach composed of static and dynamic service latency optimization. At static stage, Monte-Carlo simulation with integer-linear-programming technique is adopted to find the optimal computation offloading mapping and task backup number. At dynamic stage, a backup-adaptive dynamic mechanism is developed to avoid redundant data transmissions and executions for achieving additional energy savings and service latency enhancement. Experimental results show that our solution is able to reduce system service latency by up to 18.3% compared with representative baseline solutions.

KEYWORDS

cyber-physical systems, edge-cloud computing, energy, reliability, service latency minimization

1 | INTRODUCTION

Cyber-physical systems (CPS) are a kind of systems that deeply intertwine physical objects and software components through gathering smart sensing, computation, control and networking techniques.[1,2] In recent years, the advance in information technology has dynamically fueled the deployment of numerous emerging CPS applications such as autonomous automobile systems, healthcare monitoring, and process control systems. For these CPS applications, service latency is an utmost design concern for the sake of providing high quality-of-experience to terminal users. Edge-cloud

computing,[3] integrating both edge computing and cloud computing, is regarded as a promising computation paradigm to achieve low service latency for terminal users in CPS.

Recently, many research works have denoted to the design of latency-aware edge-cloud computing methods suitable for CPS applications. For example, Cao et al.[4] conducted the study on how to deploy heterogeneous edge servers for minimizing the service latency of the overall and individual base stations. Ghobaei-Arani et al.[5] developed a moth-flame optimization strategy based task scheduling algorithm to minimize the task execution and transfer latency. An edge computing assisted health monitoring system is designed by Sood et al.[6] to realize the goal of monitoring and analyzing users' health statistics in a real-time manner. Mudassar et al.[7] proposed a decentralized method to optimize system service latency. The basic idea of the method is to cluster edge devices into several groups and split resource intensive tasks into several subtasks such that these subtasks can execute on edge devices in parallel. A Bluetooth 5 powered architecture was demonstrated by Fraga-Lamas et al.,[8] which not only has the ability to quickly response to critical events but also can handle computing-intensive complex tasks. Jiang et al.[9] focused on reducing the execution time of artificial intelligent applications on hybrid CPU and field-programmable-gate-array platforms. However, all the above research works[4-9] fail to take into account the energy budgets and reliability requirements in designing CPS algorithms.

In addition to service latency optimization, energy minimization and reliability enhancement for CPS applications are also two hot topics in both academia and industry. From the point of energy minimization, Tariq et al.[10] investigated the issue of energy and contention-aware scheduling of precedence and deadline constrained tasks executed on edge computing devices. To deal with this issue, the authors proposed an earliest-edge-consistent-deadline-first scheduling algorithm associated with the technique of energy gradient decent voltage scaling. Peng et al.[11] put forward an enhanced nondominated sorting genetic algorithm based computation offloading mechanism to jointly optimize energy optimization and service latency. Zeng et al.[12] investigated the way of leveraging energy generation diversity to realize energy efficient service composition for CPS applications powered by green energy. From the point of reliability enhancement, Odonovan et al.[13] devised an industrial CPS architecture involving production-ready machine learning algorithms to optimize both the task processing time and the number of communication failures. A dynamic reliability prediction algorithm was delivered by Okafor[14] to accurately estimate the system-level mean time to failure. Cicirelli et al.[15] introduced a novel method to design smart CPS capable of fault tolerance with the assistance of edge computing and agent-based techniques. From the perspective of joint optimization for energy efficiency and service latency, Jiang et al.[16] demonstrated a Tabu search-based heuristic to improve the energy efficiency of real-time applications with security and reliability requirements. The main idea of the heuristic is to consecutively conduct Tabu searching procedures of task mode allocation and message security-level assignment such that an extended list scheduling method can be easily coupled into the process of producing optimal task schedule tables. Considering the stochastic durations of application execution, Jiang et al.[17] further presented an energy-efficient design of real-time and reliable applications on uniprocessor embedded systems. The authors first formulated the optimization problem into a typical multidimensional multiple-choice knapsack problem, and then leveraged the dynamic programming technique to deal with the knapsack problem. Although the above two representative research works[16,17] take into account both energy efficiency and service latency, they neglect the potential of edge-cloud framework in improving system performance.

To the best of our knowledge, most of existing latency-aware edge-cloud computing methods dedicated for CPS fail to jointly consider energy budgets and reliability requirements. Since energy budgets and reliability requirements are also key concerns in CPS, latency-aware methods neglecting energy budgets or reliability requirements may greatly degrade and damage the sustainability of CPS applications. In this article, we conduct the first study on optimizing service latency of edge-cloud computing coupled CPS under the constraints of energy budgets and reliability requirements. In summary, this article makes the following main contributions.

- We investigate the problem of service latency minimization for edge-cloud computing embedded CPS applications. In particular, we take CPS applications' energy budgets and reliability requirements into considerations throughout service latency optimization.
- We devise a methodology consisting of static and dynamic optimization algorithms. The static algorithm leverages Monte-Carlo simulation with integer-linear-programming (ILP) technique to find the optimal computation offloading mapping and task backup number. The dynamic algorithm adopts a backup-adaptive mechanism to avoid redundant task transmissions and executions at runtime.
- We carry out extensive experiments to appraise the effectivity of our solution. Experimental results reveal that our solution reduces system service latency by up to 18.3% compared with representative baseline solutions.

The rest of this article is organized as follows. Section 2 presents system models. Section 3 introduces the problem definition and outlines our proposed solution. Sections 4 and 5 describe the static and dynamic service latency optimization algorithms, respectively. The evaluation results are demonstrated in Section 6 and conclusive statements are given in Section 7.

2 | SYSTEM MODELS

This section introduces system models including system architecture model, service latency model, energy model, and reliability model.

2.1 | System architecture model

As illustrated in Figure 1, we consider typical edge-cloud computing coupled CPS consisting of numerous end users, $\mathcal{J}$ base stations $\mathcal{S}_{\text{base}} = \{\mathcal{S}^1_{\text{base}}, \mathcal{S}^2_{\text{base}}, \dots, \mathcal{S}^{\mathcal{J}}_{\text{base}}\}$, $\mathcal{K}$ heterogeneous edge servers $\mathcal{S}^{\text{edge}}_{\text{server}} = \{\mathcal{S}^{\text{edge},1}_{\text{server}}, \mathcal{S}^{\text{edge},2}_{\text{server}}, \dots, \mathcal{S}^{\text{edge},\mathcal{K}}_{\text{server}}\}$, and one cloud server $\mathcal{S}^{\text{cloud}}_{\text{server}}$. The heterogeneity of edge servers is mainly demonstrated in their computation capacity, that is, any two different edge servers exhibit different computation capacities. The computation capacity of edge server $\mathcal{S}^{\text{edge},k}_{\text{server}} \in \mathcal{S}^{\text{edge}}_{\text{server}}$ $(1 \le k \le \mathcal{K})$ is denoted by $\mathcal{P}^{\text{edge},k}_{\text{server}}$, and the computation capacity of cloud server $\mathcal{S}^{\text{cloud}}_{\text{server}}$ is represented by $\mathcal{P}^{\text{cloud}}_{\text{server}}$. Due to the consideration of deployment costs, the number of edge servers is much less than that of base stations in real world, which indicates the inequation of $\mathcal{J} > \mathcal{K}$ holds. For the sake of easy presentation, we utilize an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to describe the topological relationship among $\mathcal{J}$ base stations, $\mathcal{K}$ edge servers, and cloud server $\mathcal{S}^{\text{cloud}}_{\text{server}}$. In the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ and $\mathcal{E}$ respectively store the placement location information (e.g., latitude, longitude) and link communication information (e.g., link bandwidth, routing selection) of base stations $\{\mathcal{S}^1_{\text{base}}, \mathcal{S}^2_{\text{base}}, \dots, \mathcal{S}^{\mathcal{J}}_{\text{base}}\}$, edge servers $\{\mathcal{S}^{\text{edge},1}_{\text{server}}, \mathcal{S}^{\text{edge},2}_{\text{server}}, \dots, \mathcal{S}^{\text{edge},\mathcal{K}}_{\text{server}}\}$, and cloud server $\mathcal{S}^{\text{cloud}}_{\text{server}}$. It should point out that the mentioned topological relationship is logically full-meshed, but in fact it is a multihop network, that is, a base station is able to conduct communication with the desired edge/cloud server through other base stations.

2.2 | Service latency model

For every base station $\mathcal{S}^j_{\text{base}}$ in the concerned system, it has to offload end users' computation tasks in its charge to an edge server $\mathcal{S}^{\text{edge},k}_{\text{server}}$ or cloud server $\mathcal{S}^{\text{cloud}}_{\text{server}}$, and pray the selected edge or cloud server to process computation tasks. Given

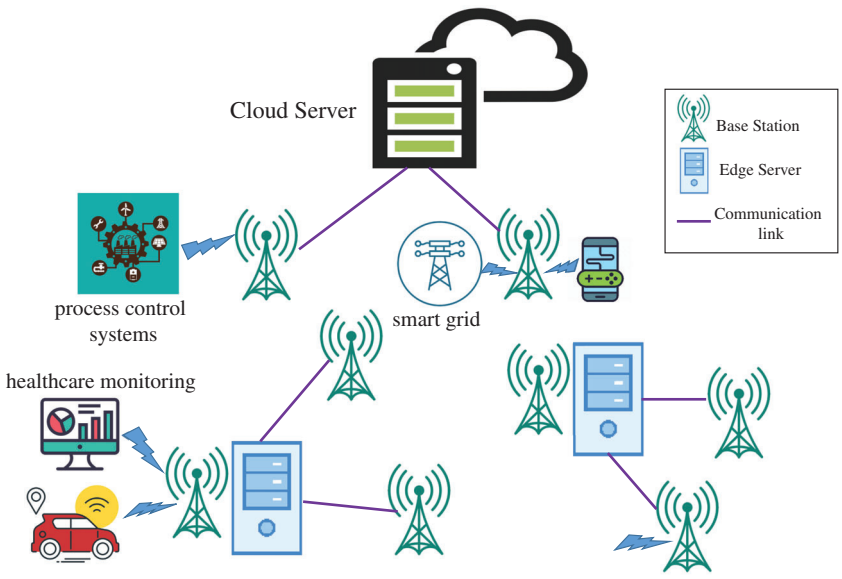


FIGURE 1 Example of edge-cloud computing assisted CPS

this, we model the service latency of base station $\mathcal{S}^j_{\text{base}}$ from two aspects of computation offloading transmission latency and execution latency. We first introduce the model of computation offloading transmission latency. To be specific, we assume that the computation tasks sent from multiple end users to base station $\mathcal{S}^j_{\text{base}}$ follow a Poisson distribution with the parameter of average arrival rate $\mathcal{R}^j_{\text{base}}$. Let $\mathcal{S}^{\text{tot}}_{\text{server}} = \{\mathcal{S}^{\text{cloud}}_{\text{server}}, \mathcal{S}^{\text{edge},1}_{\text{server}}, \mathcal{S}^{\text{edge},2}_{\text{server}}, \ldots, \mathcal{S}^{\text{edge},\mathcal{K}}_{\text{server}}\}$ denote the collection of $\mathcal{K}$ edge servers and cloud server $\mathcal{S}^{\text{cloud}}_{\text{server}}$, and $\mathcal{S}^m_{\text{server}}$ $(0 \le m \le \mathcal{K})$ represent the mth server in server set $\mathcal{S}^{\text{tot}}_{\text{server}}$. It is also assumed that the communication bandwidth between base station $\mathcal{S}^j_{\text{base}}$ and edge/cloud $\mathcal{S}^m_{\text{server}}$ is $\mathcal{C}_{j,m}$. As detailed in Reference 18, the communication latency from base station $\mathcal{S}^j_{\text{base}}$ to edge/cloud server $\mathcal{S}^m_{\text{server}}$ is calculated as

$$\mathcal{L}^{j,m}_{\text{com}} = \frac{\mathcal{D}_{j,m}}{\xi} + \frac{W_j}{\mathcal{C}_{j,m}}. \tag{1}$$

In Equation (1), $\mathcal{D}_{j,m}$ indicates the distance from base station $\mathcal{S}^j_{\text{base}}$ to edge/cloud server $\mathcal{S}^m_{\text{server}}$, ξ suggests the propagation speed of electromagnetic waves, and W_j implies the total data volume of end users' tasks at base station $\mathcal{S}^j_{\text{base}}$.

We then introduce the model of computation offloading execution latency. Specifically, we choose the widely utilized M/G/1 queue model in literature[19] to quantify execution latency of base station $\mathcal{S}^j_{\text{base}}$ connected to edge/cloud server $\mathcal{S}^m_{\text{server}}$. In this model, task execution time on edge/cloud server $\mathcal{S}^m_{\text{server}}$ is not limited to any specified probability distribution, that is, it is allowed to obey a general probability distribution function with average value μ_m and standard deviation δ_m. Note that this general probability distribution function should be given in advance before the system starts to run. Once the system is in running state, tuning this probability distribution function is prohibited. As shown in Reference 19, the execution latency of base station$\mathcal{S}^j_{\text{base}}$ connected to edge/cloud server $\mathcal{S}^m_{\text{server}}$ is calculated as

$$\mathcal{L}^{j,m}_{\text{exe}} = \frac{\mathcal{R}^j_{\text{base}}}{\mathcal{P}^m_{\text{server}}} + \frac{(\mu_m^2 + \delta_m^2)(\mathcal{R}^j_{\text{base}} + \Phi_m)}{2(\mathcal{P}^m_{\text{server}} - \mathcal{R}^j_{\text{base}} + \Phi_m)}. \tag{2}$$

In Equation (2), $\mathcal{P}^m_{\text{server}}$ denotes the computation speed supported by edge/cloud server $\mathcal{S}^m_{\text{server}}$, and Φ_m represents the sum of task arriving rates of such base stations (except base station $\mathcal{S}^j_{\text{base}}$) mapped to edge/cloud server $\mathcal{S}^m_{\text{server}}$.

Putting together the transmission latency in Equation (1) and execution latency in Equation (2), the total service latency of base station $\mathcal{S}^j_{\text{base}}$ if it establishes a connection to edge/cloud server $\mathcal{S}^m_{\text{server}}$ is expressed as

$$\mathcal{L}^{j,m}_{\text{tot}} = \frac{\mathcal{D}_{j,m}}{\xi} + \frac{W_j}{\mathcal{C}_{j,m}} + \frac{\mathcal{R}^j_{\text{base}}}{\mathcal{P}^m_{\text{server}}} + \frac{(\mu_m^2 + \delta_m^2)(\mathcal{R}^j_{\text{base}} + \Phi_m)}{2(\mathcal{P}^m_{\text{server}} - \mathcal{R}^j_{\text{base}} + \Phi_m)}. \tag{3}$$

The system service latency is then defined as the averaged service latency of all base stations in the system, that is,

$$\mathcal{L}^{\text{avg}}_{\text{sys}} = \frac{1}{J}\sum_{j=1}^{\mathcal{J}}\sum_{m=0}^{\mathcal{K}} \mathcal{A}_{j,m}\mathcal{L}^{j,m}_{\text{tot}}. \tag{4}$$

In Equation (4), $\mathcal{A}_{j,m}$ is a binary decision variable taking the value either 0 or 1. In cases where base station $\mathcal{S}^j_{\text{base}}$ decides to communicate with edge/cloud server $\mathcal{S}^m_{\text{server}}$, $\mathcal{A}_{j,m}$ takes the value 1; otherwise, $\mathcal{A}_{j,m}$ takes the value 0.

2.3 | Energy model

The whole energy dissipation of edge-cloud computing coupled CPS mainly includes two parts: one is the energy consumed by base stations to offload computation tasks from end users to edge/cloud servers, and the other is the energy consumed by edge/cloud servers to process offloaded computation tasks. Assume that the power consumption of base station $\mathcal{S}^j_{\text{base}}$ is a constant number of $\mathcal{H}^j_{\text{base}}$. The energy dissipation of base station $\mathcal{S}^j_{\text{base}}$ to transmit end users' computation tasks in its charge is therefore given by

$$E^j_{\text{base}} = \mathcal{H}^j_{\text{base}}\sum_{m=0}^{\mathcal{K}} \mathcal{A}_{j,m}\mathcal{L}^{j,m}_{\text{com}}. \tag{5}$$

The energy consumed by an edge/cloud server depends to a great extent on several functional components of processors, disks, memory, fans, and cooling systems. As pointed out in References 20-22, the processor energy dissipation accounts for a significant portion of the overall energy consumption in an edge/cloud server. Consequently, we only take the processor energy dissipation into consideration when modeling the energy dissipation of edge/cloud servers. Following the energy model in References 20-22, the energy consumed by edge/cloud server $\mathcal{S}^m_{\text{server}}$ can be estimated as

$$E^m_{\text{server}} = (\mathcal{H}^{m,\text{sta}}_{\text{server}} + \alpha_m v_m^2 \mathcal{P}^m_{\text{server}}) \sum_{j=1}^{\mathcal{J}} \mathcal{A}_{j,m} \frac{\mathcal{R}^j_{\text{base}}}{\mathcal{P}^m_{\text{server}}}, \tag{6}$$

where $\mathcal{H}^{m,\text{sta}}_{\text{server}}$ refers to the static power dissipation and it is a constant number dependent on processor architecture of edge/cloud server $\mathcal{S}^m_{\text{server}}$. α_m is also a constant number and v_m is the processor supply voltage of edge/cloud server $\mathcal{S}^m_{\text{server}}$. Combining Equations (5) and (6), the system energy consumption is naturally expressed as

$$E_{\text{sys}} = \sum_{j=1}^{\mathcal{J}} E^j_{\text{base}} + \sum_{m=0}^{\mathcal{K}} E^m_{\text{server}}. \tag{7}$$

2.4 | Reliability model

The reliability of tasks at a base station is defined as the probability that these tasks are first successfully transmitted to the target edge/cloud server without the occurrence of bit errors, and then successfully executed by the target edge/cloud server without the occurrence of soft errors. In the procedure of digital transmission, bit errors mostly arise from ambient noises, interferences, distortions, or bit synchronization errors over links. Let $\Theta^{j,m}_{\text{bit}}$ denote the constant bit error rate of the link from $\mathcal{S}^j_{\text{base}}$ to edge/cloud server $\mathcal{S}^m_{\text{server}}$, then the transmission reliability is depicted as[23]

$$\Upsilon^{j,m}_{\text{com}} = \exp(-\Theta^{j,m}_{\text{bit}} \cdot \mathcal{L}^{j,m}_{\text{com}}). \tag{8}$$

Unlike bit errors, soft errors are mainly incurred by transient faults resulting from cosmic radiations or electromagnetic interferences. Let Θ^m_{soft} represent the fault occurrence rate of edge/cloud server $\mathcal{S}^m_{\text{server}}$ on average, then it is calculated as[24,25]

$$\Theta^m_{\text{soft}} = C_m \cdot \exp(-\varpi_m \cdot \mathcal{P}^m_{\text{server}}), \tag{9}$$

where both C_m and ϖ_m are constant numbers dependent on hardware architecture. Using exponential distribution assumption, the execution reliability is therefore given by

$$\Upsilon^{j,m}_{\text{exe}} = \exp\left(-\Theta^m_{\text{soft}} \cdot \frac{\mathcal{R}^j_{\text{base}}}{\mathcal{P}^m_{\text{server}}}\right). \tag{10}$$

To meet system reliability requirements, we adopt the powerful backup technique to tolerate both bit errors and soft errors. Moreover, to check whether or not tasks are successfully processed, an acceptance test[26] is conducted after the execution of current backup on any edge/cloud server. If the acceptance test shows that there are no errors, the output results of the current backup are accepted; otherwise, they are directly thrown away. When $\mathcal{I}^{j,m}_{\text{back}}$ backups are reserved for base station $\mathcal{S}^j_{\text{base}}$, the reliability is readily derived as

$$\Upsilon^{j,m}_{\text{back}} = 1 - (1 - \Upsilon^{j,m}_{\text{com}} \cdot \Upsilon^{j,m}_{\text{exe}})^{\mathcal{I}^{j,m}_{\text{back}}}. \tag{11}$$

The system reliability is then characterized by the product of the reliabilities of all base stations in the system, that is,

$$\Upsilon^{\text{sys}}_{\text{back}} = \prod_{j=1}^{\mathcal{J}} \sum_{m=0}^{\mathcal{K}} \mathcal{A}_{j,m} \Upsilon^{j,m}_{\text{back}}. \tag{12}$$

3 | PROBLEM DEFINITION AND APPROACH OVERVIEW

This section first defines the problem of service latency optimization and then outlines our proposed approach.

3.1 | Problem definition

Our goal is to minimize system service latency by determining an optimal strategy of computation offloading and task backup for every base station under given design constraints. Given this, we define the problem of service latency optimization as follows. For the system described by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, decide the i) computation offloading mapping from base stations to edge/cloud servers and ii) number of task backups for individual base stations such that the system service latency is minimized. Five design constraints should be met in order to guarantee system scheduling feasibility. First, every base station is only allowed to forward its computation tasks to one edge/cloud server. Second, the workload of any edge/cloud server cannot exceed its maximal processing capacity. Third, the energy consumed by the whole system cannot exceed the given energy threshold. Fourth, the number of task backups for each base station cannot exceed the maximum backup number specified by the system. Finally, the system reliability with error tolerance is higher than the predefined reliability threshold. In summary, our studied optimization problem is mathematically formulated as follows.

$$\text{minimize: } \mathcal{L}_{\text{sys}}^{\text{avg}} = \frac{1}{J}\sum_{j=1}^{\mathcal{J}}\sum_{m=0}^{\mathcal{K}} \mathcal{A}_{j,m}\mathcal{L}_{\text{tot}}^{j,m} \tag{13}$$

$$\text{subject to: } \sum_{m=0}^{\mathcal{K}} \mathcal{A}_{j,m} = 1, \forall j \in [1, 2, \ \ldots \ , \mathcal{J}] \tag{14}$$

$$\sum_{j=1}^{\mathcal{J}} \mathcal{A}_{j,m}\mathcal{R}_{\text{base}}^{j} \leq \mathcal{P}_{\text{server}}^{m}, 0 \leq m \leq \mathcal{K} \tag{15}$$

$$\sum_{j=1}^{\mathcal{J}} E_{\text{base}}^{j} + \sum_{m=0}^{\mathcal{K}} E_{\text{server}}^{m} \leq E_{\text{sys}}^{\text{thr}} \tag{16}$$

$$\mathcal{I}_{\text{back}}^{j,m} \leq \mathcal{I}_{\text{back}}^{\text{thr}}, 1 \leq j \leq \mathcal{J}, 0 \leq m \leq \mathcal{K} \tag{17}$$

$$\Upsilon_{\text{back}}^{\text{sys}} = \prod_{j=1}^{\mathcal{J}}\sum_{m=0}^{\mathcal{K}} \mathcal{A}_{j,m}\Upsilon_{\text{back}}^{j,m} \geq \Upsilon_{\text{back}}^{\text{thr}}. \tag{18}$$

Equation (14) ensures that base station S_{base}^{j} is precisely mapped to one and only one edge/cloud server. Equation (15) ensures the satisfaction of maximal processing capacity constraint for each edge/cloud server. Equation (16) ensures the satisfaction of the energy upper-bound constraint. Equation (17) ensures the satisfaction of backup number constraint, where $\mathcal{I}_{\text{back}}^{\text{thr}}$ denotes the maximum backup number specified by the system. Equation (18) ensures the satisfaction of system reliability constraint, where $\Upsilon_{\text{back}}^{\text{thr}}$ denotes the predefined system reliability threshold.

3.2 | Approach overview

To tackle the problem defined in Section 3.1, we develop a two-stage approach composed of static and dynamic service latency optimization. At the static optimization stage, Monte-Carlo simulation with ILP technique is adopted to find the static computation offloading mapping and the number of task backups for each base station. We first introduce the definition of error adaptation factor for the purpose of characterizing the stochastic features of error occurrences. With the help of error adaptation factor, we then utilize the ILP technique to solve a determined optimization problem under energy consumption constraints and conduct Monte-Carlo simulations to judge whether or not the system reliability constraint is satisfied. After several attempts of tuning error adaptation factor, an optimal solution meeting both the constraints of energy consumption and system reliability can be found. Considering redundant backup transmissions and executions incurred by static optimization stage, we further propose a backup-adaptive

dynamic optimization mechanism for enhancing system service latency at runtime. At the dynamic optimization stage, once the first successful backup is detected, the transmissions and executions of other unnecessary task backups are immediately canceled. Through the above two stages, our approach can achieve the goal of minimizing system service latency.

4 | MONTE-CARLO SIMULATION WITH ILP TECHNIQUE BASED STATIC OPTIMIZATION

This section presents our Monte-Carlo simulation with ILP technique based static optimization mechanism. We first describe the main idea of our proposed strategy, and then exhibits the algorithm pseudo-code of the optimization strategy.

4.1 | Stochastic static strategy

As aforementioned, we adopt the popular backup technique to tolerate bit errors and soft errors. In best cases where no errors occur during the computation forwarding and processing procedure for base station $\mathcal{S}^{j}_{\text{base}}$, it is clearly that no redundant backup is needed to provide error tolerance. Then, let $\mathcal{R}^{j,\%}_{\text{base}}$ be the average arrival rate of base station $\mathcal{S}^{j}_{\text{base}}$ at that moment. On the contrary, in worst cases a total of $\mathcal{I}^{\text{thr}}_{\text{back}}$ backups should be completely finished for base station $\mathcal{S}^{j}_{\text{base}}$. Let $\mathcal{R}^{j,\mathcal{W}}_{\text{base}}$ be the average arrival rate of base station $\mathcal{S}^{j}_{\text{base}}$ at worst cases, then it is given by $\mathcal{R}^{j,\mathcal{W}}_{\text{base}} = \mathcal{I}^{\text{thr}}_{\text{back}} \cdot \mathcal{R}^{j,\%}_{\text{base}}$. Evidently, both $\mathcal{R}^{j,\%}_{\text{base}}$ and $\mathcal{R}^{j,\mathcal{W}}_{\text{base}}$ are constant for every base station $\mathcal{S}^{j}_{\text{base}}$. Nevertheless, in common cases the average arrival rate $\mathcal{R}^{j}_{\text{base}}$ of base station $\mathcal{S}^{j}_{\text{base}}$ is a random variable because of the stochastic characteristic of error occurrences. Therefore, we define an error adaptation factor, denoted by $\Phi \in [0, 1]$, to portray the uncertainty in average arrival rate owing to the occurrences of bit errors and soft errors. Using error adaptation factor, the average arrival rate with error tolerance from base station $\mathcal{S}^{j}_{\text{base}}$ to edge/cloud server $\mathcal{S}^{m}_{\text{server}}$ is then given by

$$\mathcal{R}^{j,m}_{\text{base}} = \Phi \times \mathcal{R}^{j,\%}_{\text{base}} + (1 - \Phi) \times \mathcal{R}^{j,\mathcal{W}}_{\text{base}}. \tag{19}$$

Recall that our objective is to minimize system service latency by determining optimal computation offloading mapping and task backup number. Given the average arrival rate $\mathcal{R}^{j,m}_{\text{base}}$ with error tolerance in Equation (19), the backup number $\mathcal{I}^{j,m}_{\text{back}}$ is facilely derived as

$$\mathcal{I}^{j,m}_{\text{back}} = \left\lceil \frac{\mathcal{R}^{j,m}_{\text{base}}}{\mathcal{R}^{j,\%}_{\text{base}}} \right\rceil. \tag{20}$$

Since the backup number $\mathcal{I}^{j,m}_{\text{back}}$ can be calculated by using Equation (20) under given error adaptation factor, the remaining question is how to decide an optimal computation offloading mapping. To tackle this issue, we propose an effective computation offloading mapping approach based on Monte-Carlo simulation with ILP technique. Specifically, we first adopt the ILP technique to derive optimal $\mathcal{A}_{j,m}$ for individual base stations under current error adaptation factor Φ, where the linear objective is Equation (13) and the linear constraints are Equations (14)-(17). Next, we generate bit errors for each communication link and soft errors for every edge/cloud server based on the probability distributions of error occurrences. Then, the system reliability corresponding to current bit errors and soft errors is effortlessly derived by using Equation (12). The above two steps produce one sample of the Monte-Carlo simulation, and enough Monte-Carlo samples are thereupon taken by repeating this process. The system reliability corresponding to numerous Monte-Carlo samples can be safely estimated as the ratio of the number of feasible samples satisfying Equation (18) to the total number of Monte-Carlo samples. If this system reliability is no less than the predefined reliability, the resultant computation offloading mapping variable $\mathcal{A}_{j,m}$ is outputted. Otherwise, we tune the current value of error adaptation factor, and repeat the procedure of Monte-Carlo simulation with ILP technique until the first feasible computation offloading mapping solution is found.

4.2 | Stochastic static algorithm

Algorithm 1. Stochastic static strategy

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

1: $\Upsilon_{back}^{sys} \leftarrow 0$;
2: $\Phi_{start} \leftarrow 0, \Phi_{end} \leftarrow 1$;
3: **while** $(\Upsilon_{back}^{sys} - \Upsilon_{back}^{thr}) \geq \xi \geq 0$ **do**
4: $\Phi \leftarrow \Phi_{start} + (\Phi_{end} - \Phi_{start})/2$;
5: **for** $j = 1$ to $\mathcal{J}$ **do**
6: **for** $m = 0$ to $\mathcal{K}$ **do**
7: determine backup number $\mathcal{I}_{back}^{j,m}$ using Eq. (20);
8: **end for**
9: **end for**
10: use an ILP solver in[27] to deal with the ILP omitting the reliability constraint (18): objective (13), subject to (14)-(17);
11: derive system reliability Υ_{back}^{sys} using Monte-Carlo simulations;/* judge whether or not the reliability constraint (18) is satisfied */
12: **if** $\Upsilon_{back}^{sys} < \Upsilon_{back}^{thr}$ **then**
13: $\Phi_{start} \leftarrow \Phi + 1$;
14: **else**
15: $\Phi_{end} \leftarrow \Phi - 1$;
16: **end if**
17: **end while**
18: **return** i) static computation offloading mapping; ii) the number of task backups for each base station.

Algorithm 1 presents the pseudo-code of our stochastic offloading strategy at static optimization stage. The input to this algorithm is the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Lines 1–2 of the algorithm initialize three variables of system reliability Υ_{back}^{sys}, the lower bound Φ_{start}, and the upper bound Φ_{end} of error adaptation factor Φ. Lines 3–17 show the iteration process of finding the optimal solution to static computation offloading mapping and the number of data backups for individual base stations. To be specific, line 3 judges whether or not the diffidence between current system reliability Υ_{back}^{sys} and system reliability threshold Υ_{back}^{thr} is larger than yet close to a sufficiently small constant ξ. If the answer is no, line 4 then randomly picks up the value of error adaptation factor Φ from its domain using binary search method. For each base station $\mathcal{S}_{base}^{j}$ mapped to edge/cloud server $\mathcal{S}_{server}^{m}$, lines 5-9 calculate backup number $\mathcal{I}_{back}^{j,m}$ using Equation (20) under the current setting of error adaptation factor Φ. Line 10 adopts an ILP solver developed in Reference 27 to address the ILP program with objective (13) and constraints (14)–(17). Based on the results obtained from line 10, line 11 derives current system reliability Υ_{back}^{sys} using Monte-Carlo simulations. Lines 12–16 update the lower bound Φ_{start} or the upper bound Φ_{end} of error adaptation factor Φ according to the relationship between system reliability Υ_{back}^{sys} and reliability threshold Υ_{back}^{thr}. Lines 18 outputs the optimal computation offloading mapping and the number of task backups for each base station at static optimization stage.

5 | BACKUP-ADAPTIVE DYNAMIC OPTIMIZATION

This section shows our backup-adaptive dynamic optimization mechanism for enhancing system service latency at runtime. We first describe the main idea of our dynamic optimization mechanism, and then exhibits the algorithm pseudo-code of the dynamic optimization mechanism.

5.1 | Backup-adaptive strategy

As early mentioned in Section 2.4, we adopt the popular task backup technique to tolerate bit errors and soft errors for the purpose of satisfying system reliability requirements. Task backup technique has a strong power in dealing with varied errors, but it inevitably increases system service latency due to the redundant backup transmissions and executions. For example, even though the first backup of any task is successfully processed without the occurrence of both bit errors and soft errors, task backup technique still toughly allows the unnecessary transmissions and executions of remaining backups. Evidently, the successful transmission and execution of only one task backup is sufficient to ensure the processing result correctness. Given this, we propose a backup-adaptive dynamic strategy at online stage. At this stage, once the first successful backup is detected by using acceptance test method, the transmissions and executions of other task backups are canceled for enhancing system service latency. We describe the pseudo-code of our backup-adaptive dynamic strategy in the next subsection.

Algorithm 2. Backup-adaptive dynamic strategy

Input:i) static computation offloading mapping; ii) the number of task backups for eachbase station.

1: **for** $j = 1$ to $\mathcal{J}$ **do**
2: **for** $m = 0$ to $\mathcal{K}$ **do**
3: **if** $\mathcal{A}_{j,m} == 1$ **then**
4: **for** $i = 1$ to $\mathcal{I}^{j,m}_{\text{back}}$ **do**
5: check whether or not the i-th backup is successfully processed using acceptance test method;
6: **if** i-th backup is successfully processed **then**
7: update $\mathcal{I}^{j,m}_{\text{back}}$ using $\mathcal{I}^{j,m}_{\text{back}} \leftarrow i$;
8: **break**;
9: **else**
10: **continue**;
11: **end if**
12: **end for**
13: **end if**
14: **end for**
15: **end for**
16: **return** updated task backup number of individual base stations.

5.2 | Backup-adaptive algorithm

Algorithm 2 shows the pseudo-code of our backup-adaptive strategy at dynamic optimization stage. The algorithm takes inputs of static computation offloading mapping and the number of task backups for each base station. Lines 1–3 of the algorithm check whether or not base station $\mathcal{S}^{j}_{\text{base}}$ is mapped to edge/cloud server $\mathcal{S}^{m}_{\text{server}}$. If the answer is yes, lines 4 and 5 then attempt to find the first successful transmission and execution backup from all task backups for base station $\mathcal{S}^{j}_{\text{base}}$. In the case where ith backup is successfully processed, lines 6–8 first update the backup number base station $\mathcal{S}^{j}_{\text{base}}$, and then exist the iteration searching process. Otherwise, lines 9 and 10 continue the iteration searching process by jumping back to line 4. The algorithm outputs the updated task backup number of individual base stations in line 16.

6 | NUMERICAL RESULTS

6.1 | Experimental configurations

In order to evaluate the effectiveness of our reliable edge-cloud computing solution, we carry out extensive experiments based on the base station database from Shanghai Telecom.[28-30] Figure 2 exhibits the location distributions of 3233 base stations in this base station database, where the number in every circle marked by red color refers to the number of base

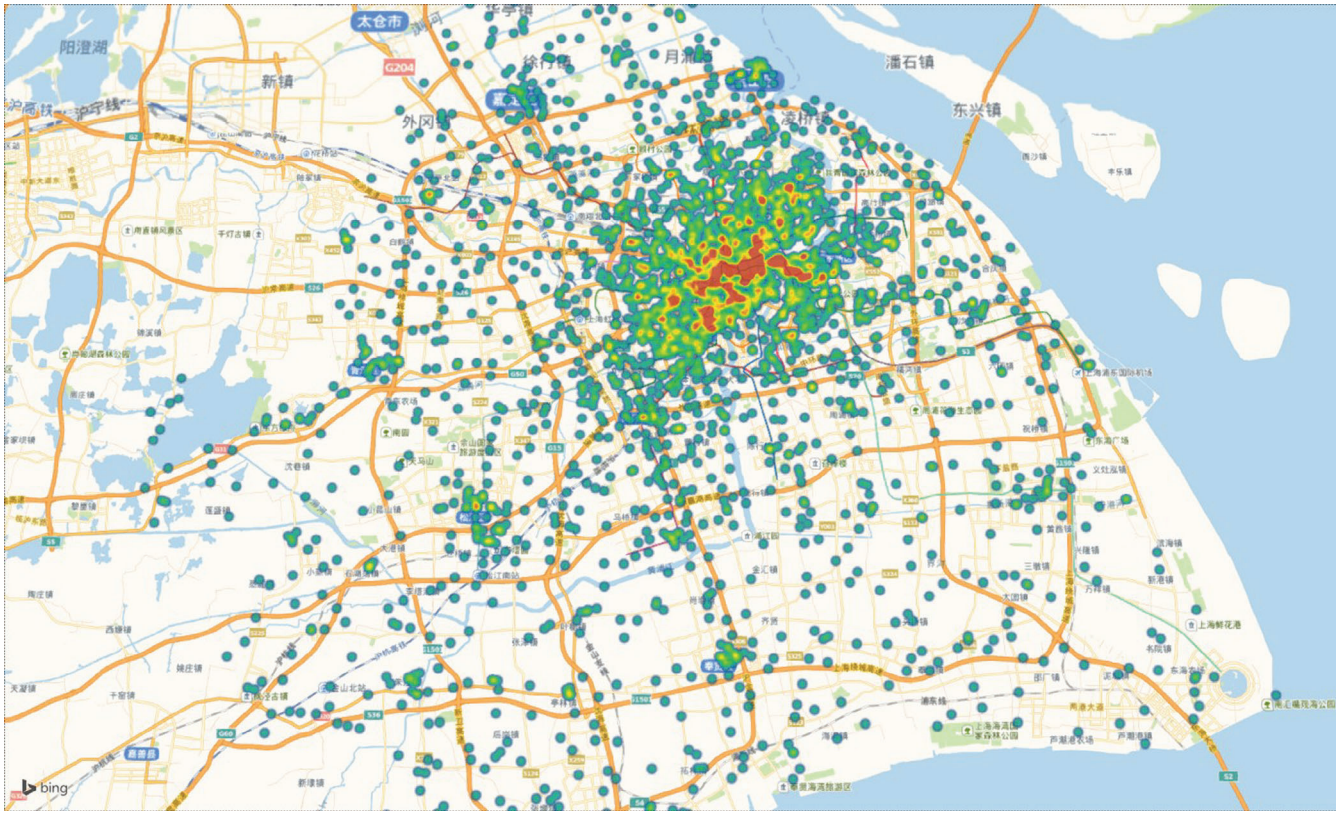


FIGURE 2 Illustration of 3233 base stations' deployment locations provided by Shanghai Telecom[28-30]

stations that have already been properly deployed in this zone. For every base station, we set its task arriving rate and data volume on average to the intervals of $[4\times10^6, 6\times10^8]$ and [1, 100] Mb, respectively.[21,31-34] In addition, we construct a set of heterogeneous edge/cloud servers based on five kinds of commercial servers in the real world. The first kind of servers comes from Microsoft Azure China (Shanghai).[35] We randomly choose a server containing ten processor cores with per core operating frequency of 3.6 GHz from such kind of servers, and ask this server to play the role of cloud server $\mathcal{S}^0_{\text{server}}$. The second kind of servers is built on the HPE ProLiant MicroServer Gen10 servers,[36] and each server contains four processor cores with per core operating frequency of 3.4 GHz. The third kind of servers is built on the Dell R230 servers,[37] and each server contains six processor cores with per core operating frequency of 3.0 GHz. The fourth kind of servers is built on the Lenovo TS250 servers,[38] and each server contains two processor cores with per core operating frequency of 3.9 GHz. The fifth kind of servers is built on the Inspur NP3020 servers,[39] and each server contains four processor cores with per core operating frequency of 3.0 GHz.

Furthermore, we utilize the later four kinds of servers to construct a set of heterogeneous edge servers. The number of each kind of edge servers is equally set to 50, and thus the size of edge server set is 200. The task execution time on every kind of edge servers is assumed to follow normal distributions, and the mean-variance parameters of the normal distributions are orderly set to the pairs of (20, 5), (14, 8), (35, 15), and (17, 10). The location distribution of edge servers is produced in a random way with the assumption that every edge server is strictly colocated with a selected base station. For each link between a base station and an edge/cloud server, its communication capacity is deemed to be in the interval of [100, 1000] KB/s.[40] The electromagnetic wave propagating rate takes the value of 2×10^5 km/s.[18]

6.2 | Experimental results

Figure 3 demonstrates system service latency achieved by three solutions under fixed edge server locations yet varied base station workloads. Note that every data point in this figure is an average of 100 simulation experiments. We are

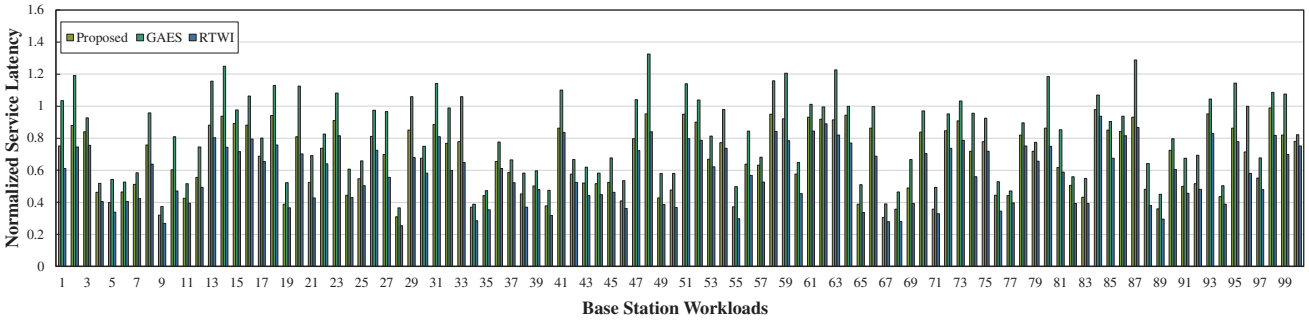


FIGURE 3 The system service latency under fixed edge server locations and varied base station workloads

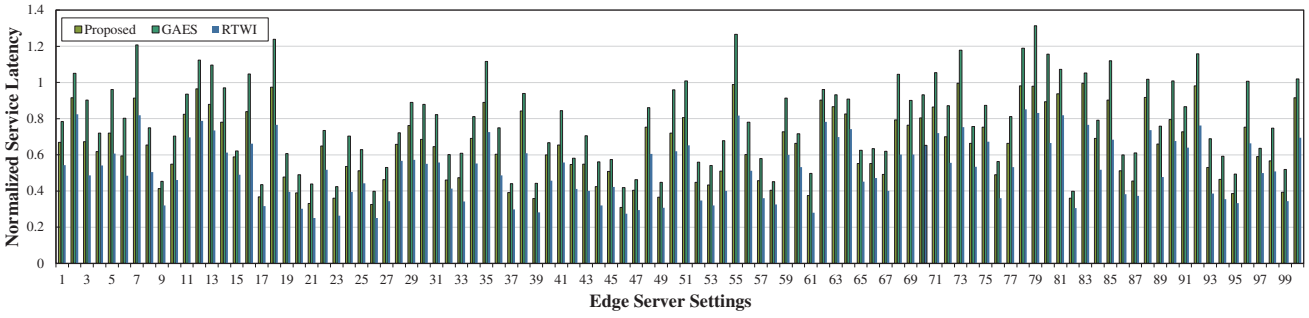
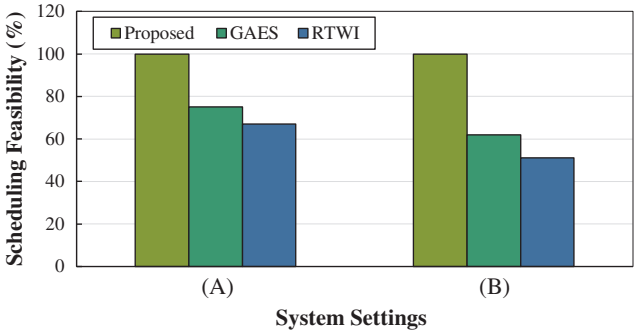


FIGURE 4 The system service latency under varied settings of edge server locations and fixed base station workloads

FIGURE 5 The task scheduling feasibility achieved by three solution with different system configurations: (A) fixed edge server locations and varied base station workloads; (B) varied settings of edge server locations and fixed base station workloads



able to observe that our proposed edge-cloud computing solution shortens service latency by up to 18.3% compared with baseline solution GAES[11] under comparison. Baseline solution GAES[11] is an enhanced nondominated sorting genetic algorithm based computation offloading mechanism to jointly optimize energy optimization and service latency. This method fails to take task reliability constraints into account. In addition, we can also see from the figure that our proposed edge-cloud computing solution is inferior to baseline solution RTWI[4] in terms of service latency, with an average gap of 13.2%. Baseline solution RTWI[4] aims to minimize not only the average response time of all base stations but also the response time of each base station. However, it fails to consider the constraints of energy budgets and reliability requirements.

Figure 4 presents the system service latency achieved by three solutions under fixed base station workloads yet varied edge server locations. Similar to Figure 3, every data point in this figure is also an average of 100 simulation experiments. As observed from the figure, the system service latency achieved by our proposed solution is 17.4% smaller than that of baseline solution GAES[11] but 19.1% higher than that of baseline solution RTWI.[4] This is mainly because our proposed solution allows multiple executions of a same task to provide the desired fault tolerance requirements, but the baseline solution RTWI[4] ignores the fault tolerance requirements and one task is exactly executed once even occurring bit errors or soft errors. Figure 5 plots the task scheduling feasibility achieved by our proposed solution and two benchmarking

solutions. The task scheduling feasibility is given by the ratio of the number of simulations in which tasks are successfully scheduled under constraints of energy budgets and reliability requirements to the whole number of simulations under test (i.e., 10,000). The results in this figure clearly exhibit that our proposed solution can maintain 100% task scheduling feasibility whereas neither of the two benchmarking solutions can guarantee task scheduling feasibility. This is due to the fact that our proposed solution takes into account both energy budgets and reliability requirements, while benchmarking solutions neglect energy and/or reliability constraints.

7 | CONCLUSION

Emerging CPS applications like autonomous automobile systems, healthcare monitoring and process control systems, desire to reduce service latency for providing high quality-of-experience to terminal users. In this article, we propose a two-stage method to tackle the problem of minimizing service latency of edge-cloud computing coupled CPS with considerations of the energy budgets and reliability requirements. The static stage aims to find the optimal computation offloading mapping and the number of task backups while the dynamic stage strives for avoiding redundant task transmissions and executions at runtime. Extensive experimental results reveal that our method reduces system service latency by up to 18.3% while guaranteeing the satisfaction of specific energy budgets and reliability requirements.

ACKNOWLEDGMENT

This work was partially supported by Key R&D Program of Guangdong Province under Grant 2019B010136003, National Key Research and Development Program of China under Grant 2018YFB2101300, and Natural Science Foundation of China under Grant 61872147.

ORCID

Kun Cao https://orcid.org/0000-0003-4872-4908
Keqin Li https://orcid.org/0000-0001-5224-4048

REFERENCES

1. Hassan M, Rehmani M, Chen J. Differential privacy techniques for cyber physical systems: a survey. *IEEE Commun Surv Tutor*. 2019;22(1):746-789.
2. Zhao J, Zhu M, Li X, Huang Z, Li J, Song J. Solving Boolean polynomial systems by parallelizing characteristic set method for cyber-physical systems. *Softw Pract Exper*. 2020;1-1.
3. Rodrigues T, Suto K, Nishiyama H, Kato N. Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control. *IEEE Trans Comput*. 2016;66(5):810-819.
4. Cao K, Li L, Cui Y, Wei T, Hu S. Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing. *IEEE Trans Ind Inform*. 2021;17(1):494-503.
5. Ghobaei-Arani M, Souri A, Safara F, Norouzi M. An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing. *Trans Emerg Telecommun Technol*. 2020;31(2):1-14.
6. Sood S, Mahajan I. Fog-cloud based cyber-physical system for distinguishing, detecting and preventing mosquito borne diseases. *Futur Gener Comput Syst*. 2018;88:764-775.
7. Mudassar M, Zhai Y, Liao L, Zahid M, Afzal F. Decentralized and secure cooperative edge node grouping to process IoT applications in heterogeneous smart cyber-physical systems. Paper presented at: Proceedings of the IEEE International Bhurban Conference on Applied Sciences and Technology; 2020:395-400.
8. Fraga-Lamas P, Lopez-Iturri P, Celaya-Echarri M, et al. Design and empirical validation of a Bluetooth 5 fog computing based industrial CPS architecture for intelligent industry 4.0 shipyard workshops. *IEEE Access*. 2020;8:45496-45511.
9. Jiang W, Song Z, Zhan J, He Z, Wen X, Jiang K. Optimized co-scheduling of mixed-precision neural network accelerator for real-time multitasking applications. *J Syst Archit*. 2020;110:1-12.
10. Tariq U, Ali H, Liu L, Panneerselvam J, Zhai X. Energy-efficient static task scheduling on VFI-based NoC-HMPSoCs for intelligent edge devices in cyber-physical systems. *ACM Trans Intell Syst Technol*. 2019;10(6):1-22.
11. Peng K, Huang H, Pan W, Wang J. Joint optimisation for time consumption and energy consumption of multi-application and load balancing of cloudlets in mobile edge computing. *IET Cyber-Phys Syst Theory Appl*. 2020;5(2):196-206.
12. Zeng D, Gu L, Yao H. Towards energy efficient service composition in green energy powered cyber-physical fog systems. *Futur Gener Comput Syst*. 2020;105:757-765.
13. Odonovan P, Gallagher C, Bruton K, OSullivan D. A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications. *Manuf Lett*. 2018;15:139-142.
14. Okafor K. Dynamic reliability modeling of cyber-physical edge computing network. *Int J Comput Appl*. 2019;1-11.

15. Cicirelli F, Fortino G, Guerrieri A, Spezzano G, Vinci A. Edge enabled development of smart cyber-physical environments. Paper presented at: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics; 2016:3463-3468.
16. Jiang W, Pop P, Jiang K. Design optimization for security-and safety-critical distributed real-time applications. *Microprocess Microsyst*. 2017;52:401-415.
17. Jiang W, Pan X, Jiang K, Wen L, Dong Q. Energy-aware design of stochastic applications with statistical deadline and reliability Guarantees. *IEEE Trans Comput-Aid Des Integrat Circuits Syst*. 2019;38(8):1413-1426.
18. Miller K. Communication Theories: Perspectives, Processes, and Contexts. McGraw-Hill 2005.
19. Fuhrmann S, Cooper R. Stochastic decompositions in the M/G/1 queue with generalized vacations. *Oper Res*. 1985;33(5):1117-1129.
20. Cao K, Zhou J, Xu G, Wei T, Hu S. Exploring renewable-adaptive computation offloading for hierarchical QoS optimization in fog computing. *IEEE Trans Comput Aid Design Integr Circuits Syst*. 2020;39(10):2095-2108.
21. Cao K, Xu G, Zhou J, Wei T, Chen M, Hu S. QoS-adaptive approximate real-time computation for mobility-aware IoT lifetime optimization. *IEEE Trans Comput Aid Design Integr Circuits Syst*. 2019;38(10):1799-1810.
22. Li K, Tang X, Li K. Energy-efficient stochastic task scheduling on heterogeneous computing systems. *IEEE Trans Parall Distrib Syst*. 2014;25(11):2867-2876.
23. Li L, Cong P, Cao K, et al. Game theoretic feedback control for reliability enhancement of EtherCAT-based networked systems. *IEEE Trans Comput Aid Design Integr Circuits Syst*. 2019;38(9):1599-1610.
24. Cao K, Zhou J, Wei T, Chen M, Hu S, Li K. A survey of optimization techniques for thermal-aware 3D processors. *J Syst Archit*. 2019;97:397-415.
25. Li L, Zhou J, Wei T, Chen M, Hu X. Learning-based modeling and optimization for real-time system availability. *IEEE Trans Comput*. 2020;1-1.
26. Haque M, Aydin H, Zhu D. On reliability management of energy-aware real-time systems through task replication. *IEEE Trans Parall Distrib Syst*. 2017;28(3):813-825.
27. Lin S, Schutter B, Xi Y, Hellendoorn H. Fast model predictive control for urban road networks via MILP. *IEEE Trans Intell Transp Syst*. 2011;12(3):846-856.
28. Wang S, Guo Y, Zhang N, Yang P, Zhou A, Shen X. Delay-aware microservice coordination in mobile edge computing: a reinforcement learning approach. *IEEE Trans Mob Comput*. 2019;1–1.
29. Wang S, Zhao Y, Xu J, Yuan J, Hsu C. Edge server placement in mobile edge computing. *J Parall Distrib Comput*. 2019;127:160-168.
30. The distribution of 3233 base stations from Shanghai Telecom. http://www.sguangwang.com/dataset/telecom.zip. Accessed 1 January 2020.
31. Jayaseelan R, Mitra T. Temperature aware task sequencing and voltage scaling. Paper presented at: Proceedings of the IEEE International Conference on Computer-Aided Design; 2008:618-623.
32. Cao K, Zhou J, Cong P, et al. Affinity-driven modeling and scheduling for makespan optimization in heterogeneous multiprocessor systems. *IEEE Trans Comput Aid Design Integr Circuits Syst*. 2019;38(7):1189-1202.
33. Zhou J, Sun J, Cong P, et al. Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT. *IEEE Trans Serv Comput*. 2020;13(4):745-758.
34. Zhou J, Sun J, Zhang M, Ma Y. Dependable scheduling for real-time workflows on cyber-physical cloud systems. *IEEE Trans Ind Inform*. 2020;1-1.
35. Microsoft Azure China (Shanghai). https://azure.microsoft.com/en-us/global-infrastructure/china/. Accessed 6 January 2020.
36. HPE ProLiant MicroServer Gen10 server. https://buy.hpe.com/b2c/us/en/servers/proliant-microserver/. Accessed 13 January 2020.
37. Dell PowerEdge R230 server. https://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Docum%ents/Dell_PowerEdge_R230_SpecSheet_final.pdf. Accessed 3 February 2020.
38. Lenovo ThinkServer TS250. http://b2b.lenovo.com.cn/dcgProduct/server/ts250.html. Accessed 7 February 2020.
39. Inspur NP3020 servers. http://www.szinspur.com/TowerServer/NP3020/. Accessed 10 February 2020.
40. Behr Technologies Inc 6 leading types of IoT wireless technology and their best use cases; 2019.

How to cite this article: Cao K, Wei T, Chen M, Li K, Weng J, Tan W. Exploring reliable edge-cloud computing for service latency optimization in sustainable cyber-physical systems. *Softw Pract Exper*. 2021;51:2225–2237. https://doi.org/10.1002/spe.2942