Reliability-Driven End–End–Edge Collaboration for Energy Minimization in Large-Scale Cyber-Physical Systems

Kun Cao, *Member, IEEE*, Jian Weng, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—In recent years, cyber-physical systems (CPS) have been widely deployed in industrial manufacturing fields and our daily living domains. End–end–edge collaboration, coupling mobile edge computing and device-to-device communication, is a promising computation paradigm to meet the stringent real-time demands of large-scale CPS applications. However, energy and reliability concerns should be carefully addressed in end–end–edge collaboration-empowered large-scale CPS due to the limited energy supply and inherent openness characteristic of end devices. In this article, we explore the reliability-driven energy optimization of end–end–edge collaborated large-scale CPS applications. We develop a reliability-driven end–end–edge collaboration approach to deal with the energy minimization problem. Our approach first designs a clustering method to quantify differentiated energy demands by analyzing the energy dissipation composition of heterogeneous applications. Afterward, our approach leverages incremental control and swarm intelligence-based techniques to obtain energy-efficient reliability-guaranteed task offloading solutions for differentiated application clusters. Experimental results reveal that our approach achieves 51.48% energy savings compared with peer algorithms.**

***Index Terms*—Cyber-physical systems (CPSs), device-to-device (D2D) communication, energy, mobile edge computing (MEC), reliability.**

I. Introduction

CYBER-PHYSICAL systems (CPSs) are the tight integration of hardware and software components by leveraging advanced networking, sensing, computing, controlling, and communication technologies [1]. Nowadays, the spatial and temporal dimensions of the CPS have been continuously increasing due to the proliferation of interconnected end devices. A variety of large-scale CPS applications, e.g., smart grid, industrial control systems, intelligent transportation, and personalized healthcare, have been widely deployed across diverse domains. Oftentimes, these CPS applications have stringent real-time requirements since delayed outputs may incur unacceptable timing faults [2], [3].

Recently, an appealing paradigm of end–end–edge collaboration [4] coupling mobile edge computing (MEC) and device-to-device (D2D) communication techniques has attracted widespread attention. In the common MEC paradigm, real-time tasks on an individual end device are able to complete their execution locally or offload computation instructions to MEC servers. However, this computing paradigm cannot well handle the heterogeneity of end devices during task offloading procedures, resulting in unbalanced resource utilization among end devices in a local network [5], [6]. Unlike the MEC paradigm, the D2D communication technique permits end devices with high resource utilization to request nearby end devices with underutilized resources to facilitate task execution. Consequently, MEC and D2D communication techniques are complementary to each other, which inspires us to develop the end–end–edge collaborated task offloading method. In this integrated paradigm, real-time tasks on an end device can offload their computation to either MEC servers or adjacent end devices for accomplishment. To fulfill task real-time demands, it is natural to incorporate the end–end–edge collaboration method into the design of large-scale CPS.

Nevertheless, energy management should be carefully conducted due to the economical deployment and maintenance concerns in the large-scale CPS. To this end, considerable works [7], [8], [9], [10] have been devoted to developing energy-aware end–end–edge collaboration solutions. For instance, Cao et al. [7] developed an energy-efficient framework based on convex optimization techniques to minimize the energy consumption for both binary and partial CPS computation offloading scenarios. Kai et al. [8] devised a two-stage method to optimize the number of executed tasks (i.e., system throughput) under energy constraints. Yang et al. [9] proposed a game theoretic offloading approach to jointly optimize the response latency and energy dissipation of independent tasks. Leveraging the alternating direction method of multipliers algorithm, Sun et al. [10] aimed to minimize the task response latency under energy budget constraints. However, all the aforementioned works [7], [8], [9], [10] fail to take the intertask dependence into energy optimization.

Manuscript received 9 July 2022; revised 18 April 2023; accepted 7 July 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62102164, in part by the China Postdoctoral Science Foundation under Grant 2021T140272 and Grant 2021M691240, in part by the Science and Technology Project of Guangzhou under Grant 202201010573, and in part by the Fundamental Research Funds for the Central Universities under Grant 21621025. Associate Editor: Y.-K. Lin. *(Corresponding author: Kun Cao.)*

Kun Cao and Jian Weng are with the College of Information Science and Technology, Jinan University, Guangzhou 510632, China (e-mail: kuncao@jnu.edu.cn; cryptweng@gmail.com).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TR.2023.3297124.

Digital Object Identifier 10.1109/TR.2023.3297124

TABLE I
COMPARISON OF OUR WORK WITH RELATED WORKS IN LITERATURE

Literature	Energy	Latency	Reliability	Type	Independent task	Single DAG	Multi-DAG	MEC	D2D	E3C
Sun et al. [4]	✓	✓			✓			✓	✓	✓
Pu et al. [5]	▲	✓			✓				✓	
Wang et al. [6]	▲	✓			✓			✓		
Cao et al. [7]	▲	✓			✓			✓	✓	✓
Kai et al. [8]	✓	✓			✓			✓	✓	✓
Yang et al. [9]	▲	▲			✓			✓	✓	✓
Sun et al. [10]	✓	▲			✓			✓	✓	✓
Naithani et al. [11]	✓	✓	▲	∗	–	–	–			
Ansari et al. [12]	▲	✓	✓	∗	✓					
Li et al. [13]		✓	▲	♮	✓					
Savino et al. [14]	✓	✓	▲	∗	–	–	–			
Hu et al. [15]	▲	✓	✓	∗		✓				
Cao et al. [16]	▲	✓	✓	♮	–	–	–	✓		
This article	▲	✓	✓	♮			✓	✓	✓	✓

▲: Optimization objective; ✓: Key concerns / constraints; ∗: Soft errors ; ♮: Soft and bit errors ; Multi-DAG: Multiple DAG applications; D2D: Device-to-device communication; E3C: End-end-edge collaboration; –: Not mentioned; and Blank space: Not considered in related works.

In addition to energy management, reliability augmentation is also a hot topic in CPS environments because real-time tasks are vulnerable to both bit and soft errors arising from the inherent openness characteristic of end devices. From the perspective of reliability optimization, Naithani et al. [11] presented an online task scheduler to optimize the system overall reliability by analyzing the reliability features of running applications. Ansari et al. [12] put forward a two-stage scheme to wisely determine the optimal replicas of individual real-time tasks. Li et al. [13] demonstrated a feedback controlling method to improve the reliability of EtherCAT networks. An extremal optimization theory-based heuristic algorithm was designed by Savino et al. [14] to augment the system resiliency to soft errors. Leveraging the processor-merging technique, Hu et al. [15] exhibited an energy-efficient task scheduling approach to reduce the system energy dissipation under timing and reliability constraints. Cao et al. [16] investigated the joint optimization of response latency and processor wearout under reliability and energy constraints for large-scale CPS.

As an intuitive presentation, we compare the related works [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] in Table I. As shown in the table, existing works fail to jointly bring the end–end–edge collaboration paradigm and reliability concerns into the energy optimization of the large-scale CPS. In this article, we investigate the reliability-driven energy optimization of end–end–edge collaboration-empowered large-scale CPS. Particularly, we consider both the intertask dependence depicted by the directed acyclic graph (DAG) and the concurrent task offloading of multiple DAG applications. In summary, we make the following main contributions.

1) We design a clustering approach to quantify differentiated DAG energy demands by analyzing the energy dissipation composition of heterogeneous DAG applications.
2) We devise an incremental control-based task offloading scheme for computation-intensive DAG clusters by analyzing the computation energy optimality.
3) We develop a swarm intelligence-based task offloading scheme for communication-intensive DAG clusters by integrating a novel DAG type transformation technique.
4) We validate the performance of our approach on both synthetic and real-life DAG applications. Experimental results show that our approach achieves 51.48% energy savings compared with peer algorithms.

The rest of this article is organized as follows. Section II introduces the system architecture and models. Section III presents the problem formulation and solution overview. In Section IV, we present a DAG clustering method. In Sections V and VI, we devise task offloading schemes for computation-intensive and communication-intensive DAG clusters, respectively. We evaluate our solution in Sections VII and VIII. Finally, Section IX concludes this article.

II. SYSTEM ARCHITECTURE AND MODELS

A. System Architecture

We consider an end–end–edge collaborated large-scale CPS architecture consisting of M end devices $D = \{D_1, D_2, \cdots, D_M\}$, a base station B, and an edge server S. For each end device D_m $(1 \leq m \leq M)$, its computing capacity is depicted by a processor Θ_m with supply voltage v_m and operating frequency f_m. The edge server S is empowered by the container-based virtualization techniques that support virtual packaging and isolation of different applications [17]. Let $C = \{C_1, C_2, \ldots, C_H\}$ be a collection of total H available containers created by edge server S. Then, every container C_h $(1 \leq h \leq H)$ can be depicted by a tuple $C_h : \{b_h, f_h\}$, where b_h is the communication bandwidth and f_h is the operating frequency. Note that base station B is generally deployed near the location of edge server S, and it acts as a global governor grasping the whole system information, e.g., routing selection, workload, end device states, etc.

B. Application Model

Support that every end device D_m is associated with an application described by DAG $G_m = (\mathcal{V}_m, \mathcal{E}_m)$. $\mathcal{V}_m = \{\tau_{m,1}, \tau_{m,2}, \ldots, \tau_{m,\xi_m}\}$ is a set of total ξ_m precedence constrained real-time tasks in the application. $\mathcal{E}_m = \{(e_{m,i,p}, \eta_{m,i,p}) | 1 \leq i, p \leq \xi_m, i \neq p\}$ is a set utilized for capturing the task dependence and communication data

volume from task $\tau_{m,i}$ to task $\tau_{m,p}$. If task $\tau_{m,p}$ is a direct successor of $\tau_{m,i}$, $e_{m,i,p}$ is hence set to 1, i.e., task $\tau_{m,p}$ cannot start its execution until it receives a total amount of $\eta_{m,i,p}$ communication data from task $\tau_{m,i}$. For every task $\tau_{m,i}$, its characteristics are depicted by a tuple $\tau_{m,i}:\{\mu_{m,i}, W_{m,i}, \Upsilon_{m,i}, T_m, R_m\}$, where $\mu_{m,i} \in [0,1]$ is the task activity factor [18], $W_{m,i}$ is the number of CPU instruction cycles, $\Upsilon_{m,i}$ is the data volume of CPU instruction cycles, T_m is the common deadline, and R_m is the reliability goal. In end–end–edge collaborated systems, task $\tau_{m,i}$ is equipped with local, D2D, and remote execution modes. In the local execution mode, task $\tau_{m,i}$ should completely finish its execution on end device D_m without the assistance of edge server S or other end devices. In the D2D execution mode, task $\tau_{m,i}$ is able to offload its computation via D2D links to another end device D_π $(1 \leq \pi \leq M, \pi \neq m)$, and then, asks end device D_π to accomplish task execution. Similar to the D2D mode, the remote execution mode allows task $\tau_{m,i}$ to transmit its computation to edge server S, and then, request edge server S to perform task execution. Since base station B is a global governor, it needs to decide the task execution modes according to the system information.

C. Reliability Model

We consider the occurrence of both bit errors and soft errors. Specifically, bit errors primarily occur on communication links due to ambient interferences or bit synchronization errors [13]. Let $R^{m,i,\rho}_{\text{biterror}}$ be the capacity of task $\tau_{m,i}$ in tolerating bit errors, where $\rho \in \{m, \pi, h\}$ indicates the execution mode of task $\tau_{m,i}$. In cases of task $\tau_{m,i}$ in the local execution mode (i.e., $\rho = m$), the communication reliability $R^{m,i,\rho}_{\text{biterror}}$ is deemed to be 1. Moreover, in cases where task $\tau_{m,i}$ is offloading to end device D_π or container C_h at time instance t (i.e., $\rho = \pi$ or $\rho = h$), $R^{m,i,\rho}_{\text{biterror}}$ is given by [13]

$$R^{m,i,\rho}_{\text{biterror}} = \exp\{t \times \lambda^{m,\rho}_{\text{biterror}}\} \tag{1}$$

where $\lambda^{m,\rho}_{\text{biterror}}$ is the constant bit error rate of the communication link. Unlike bit errors, soft errors may appear during task executions on processors. Let $R^{m,i,\rho}_{\text{softerror}}$ denote the capacity of task $\tau_{m,i}$ in tolerating soft errors at time instance t, the execution reliability is then expressed as [19]

$$R^{m,i,\rho}_{\text{softerror}} = R^{m,i,\rho}_{\text{parent}} \times \exp\{t \times \lambda^{\rho}_{\text{softerror}}\} \tag{2}$$

where $\lambda^{\rho}_{\text{softerror}}$ is the constant soft error rate. $R^{m,i,\rho}_{\text{parent}}$ is the probability that the correct communication data of all direct parents is successfully delivered to task $\tau_{m,i}$. Combining (1) and (2), the reliability of task $\tau_{m,i}$ is hence inferred by

$$R^{m,i,\rho}_{\text{reliability}} = R^{m,i,\rho}_{\text{biterror}} \times R^{m,i,\rho}_{\text{softerror}}. \tag{3}$$

D. Energy Model

The energy dissipation of end devices can be decomposed into static and dynamic components [18], [20]. Let P^m_{static} be the static power of end device D_m, then the static energy consumption during one scheduling horizon T is attained by

$$E^m_{\text{static}} = P^m_{\text{static}} \times T. \tag{4}$$

The dynamic energy consumption of end device D_m depends on the execution mode of task $\tau_{m,i}$. In the local execution mode, the dynamic energy consumed by processor Θ_m when running task $\tau_{m,i}$ is depicted by [18], [20]

$$E^{m,i}_{\text{localexe}} = \psi_m \times v_m^2 \times \mu_{m,i} \times W_{m,i} \tag{5}$$

where ψ_m is the effective switch capacitance of the processor Θ_m. Meanwhile, end device D_m should deliver the output results of task $\tau_{m,i}$ to direct successors. Let $\mathcal{I}^{m,i}_{\text{child}}$ denote a collection of direct successors of task $\tau_{m,i}$, where each element $\tau_{m,p} \in \mathcal{I}^{m,i}_{\text{child}}$ has to receive a total amount of $\eta_{m,i,p}$ communication data from task $\tau_{m,i}$. Consequently, the communication energy of end device D_m when delivering the output results of task $\tau_{m,i}$ to task $\tau_{m,p}$ assumed to be executed on end device D_k $(1 \leq k \leq M)$ is inferred by [21], [22]

$$E^{m,i,p}_{\text{localcom}} = \begin{cases} P^m_{\text{d2d}} \times \eta_{m,i,p}/\vartheta_{m,k}, & \text{if } k \neq m \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where P^m_{d2d} denotes the D2D transmission power of end device D_m. $\vartheta_{m,k}$ represents the D2D communication rate between end devices D_m and D_k, which can be derived by $\vartheta_{m,k} = b_{m,k} \times \log_2(1 + P^m_{\text{d2d}} \times g_{m,k}/\omega)$ [21], [22]. $b_{m,k}$ is the communication bandwidth, $g_{m,k}$ is the channel gain, and ω stands for the background interferences. The aforementioned data transmission producer will be triggered when the destination (i.e., end device D_k) of task $\tau_{m,p}$ is determined. Apart from the local execution mode, task $\tau_{m,i}$ can also select the D2D execution mode of offloading its computation to end device D_π. Then, the energy dissipation of delivering task $\tau_{m,i}$ is calculated as

$$E^{m,i,\pi}_{\text{d2dcom}} = P^m_{\text{d2d}} \times \Upsilon_{m,i}/\vartheta_{m,\pi} \tag{7}$$

where $\vartheta_{m,\pi}$ is given by $b_{m,\pi} \times \log_2(1 + P^m_{\text{d2d}} \times g_{m,\pi}/\omega)$ [21], [22]. The energy dissipation of end device D_π when handling offloaded task $\tau_{m,i}$ is inferred by

$$E^{m,i,\pi}_{\text{d2dexe}} = P^\pi_{\text{receive}} \times T_{m,i,\pi} + \psi_\pi \times v_\pi^2 \times \mu_{m,i} \times W_{m,i} \tag{8}$$

where $T_{m,i,\pi}$ stands for the D2D communication time between edge devices D_m and D_π for task $\tau_{m,i}$, and it is given by $T_{m,i,\pi} = \Upsilon_{m,i}/\vartheta_{m,\pi}$. P^π_{receive} is the receiving power of targeting edge device D_π. Similarly, the energy dissipation of edge device D_m when transmitting task $\tau_{m,i}$ to container C_h is calculated as $E^{m,i,h}_{\text{remcom}} = P^m_{\text{remote}} \times \Upsilon_{m,i}/(b_{m,h} \times \log_2(1 + P^m_{\text{remote}} \times g_{m,h}/\omega))$, where P^m_{remote} is the transmission power of end device D_m in the remote execution mode [21], [22].

III. Problem Formulation and Proposed Approach

A. Problem Formulation

We are dedicated to minimizing the whole energy dissipation of end devices in end–end–edge collaborated large-scale CPS. We present a problem formulation based on the integer linear programming (ILP) through introducing binary variables $\alpha_{m,i}$,

$\beta_{m,i,\pi}$, $\gamma_{m,i,h}$, and $\delta_{i,j}$ as follows:

$$\alpha_{m,i} = \begin{cases} 1, & \text{if } \tau_{m,i} \text{ is in local execution mode} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$\beta_{m,i,\pi} = \begin{cases} 1, & \text{if } \tau_{m,i} \text{ is transmitted to device} D_\pi \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$\gamma_{m,i,h} = \begin{cases} 1, & \text{if } \tau_{m,i} \text{ is offloaded to container} C_h \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

$$\delta_{i,j} = \begin{cases} 1, & \text{if } \tau_{m,i} \text{ starts execution before } \tau_{n,j} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

where $\tau_{n,j}$ is the jth task in DAG G_n $(1 \le n \le M, 1 \le j \le \xi_n)$. Our ILP objective function is expressed as

$$\text{minimize } E^{\text{total}}_{\text{energy}} = \sum_{m=1}^{M} E^{m}_{\text{energy}}. \tag{13}$$

In the objective function, E^{m}_{energy} denotes the energy dissipation of end device D_m, and it is derived by

$$E^{m}_{\text{energy}} = E^{m}_{\text{static}} + \sum_{i=1}^{\xi_m} \alpha_{m,i} \times \left(E^{m,i}_{\text{localexe}} + \sum_{\tau_{m,p}} E^{m,i,p}_{\text{localcom}} \right) + \sum_{n=1,n\neq m}^{M} \sum_{j=1}^{\xi_n} \beta_{n,j,m} \times E^{n,j,m}_{\text{d2dexe}} + \sum_{i=1}^{\xi_m} \sum_{\pi=1,\pi\neq m}^{M} \beta_{m,i,\pi} \times E^{m,i,\pi}_{\text{d2dcom}} + \sum_{i=1}^{\xi_m} \sum_{h=1}^{H} \gamma_{m,i,h} \times E^{m,i,h}_{\text{remcom}}. \tag{14}$$

Meanwhile, the following linear constraints cannot be violated for the sake of generating a feasible task offloading solution.

1) Every task should satisfy its deadline constraint. Let $T^{m,i}_{\text{start}}$ be the start time of task $\tau_{m,i}$, then we have

$$T^{m,i}_{\text{start}} + \alpha_{m,i} \times \frac{W_{m,i}}{f_m} + \sum_{\pi=1,\pi\neq m}^{M} \beta_{m,i,\pi} \times \frac{W_{m,i}}{f_\pi} + \sum_{h=1}^{H} \gamma_{m,i,h} \times \frac{W_{m,i}}{f_h} \le T_m \forall \tau_{m,i}. \tag{15}$$

2) Every task is performed in only one execution mode

$$\alpha_{m,i} + \sum_{\pi=1,\pi\neq m}^{M} \beta_{m,i,\pi} + \sum_{h=1}^{H} \gamma_{m,i,h} = 1 \, \forall \tau_{m,i}. \tag{16}$$

3) Every task should meet the reliability constraint

$$R^{m,i,\rho}_{\text{reliability}} \ge R_m \forall \tau_{m,i}, \rho \in \{m, \pi, h\}. \tag{17}$$

4) The energy dissipation of every end device is upper bounded. Let E^{m}_{upper} be the threshold on energy consumption of end device D_m, then we have

$$E^{m}_{\text{energy}} \le E^{m}_{\text{upper}} \forall m \in [1, M]. \tag{18}$$

5) The intertask precedence constraint should be met. Let $T^{m,i}_{\text{finish}}$ be the finish time of task $\tau_{m,i}$, then we acquire

$$T^{m,p}_{\text{start}} \ge T^{m,i}_{\text{finish}} \times e_{m,i,p} \forall \tau_{m,i}, \tau_{m,p}. \tag{19}$$

6) All tasks are executed within their durations with no overlap. Given two tasks $\tau_{m,i}$ and $\tau_{n,j}$, and a large enough constant number $\mathcal{Z}$ (e.g., 100 000 in our experiments), the following conditions are hence held:

$$1 \le \delta_{i,j} + \delta_{j,i} \, \forall \tau_{m,i} \in \mathcal{V}_m, \tau_{n,j} \in \mathcal{V}_n \tag{20}$$

$$T^{m,i}_{\text{start}} \le T^{n,j}_{\text{start}} + (1 - \delta_{i,j}) \times \mathcal{Z} \, \forall \tau_{m,i} \in \mathcal{V}_m, \tau_{n,j} \in \mathcal{V}_n \tag{21}$$

$$T^{n,j}_{\text{start}} \le T^{m,i}_{\text{start}} + \delta_{i,j} \times \mathcal{Z} \, \forall \tau_{m,i} \in \mathcal{V}_m, \tau_{n,j} \in \mathcal{V}_n. \tag{22}$$

Moreover, $\forall \tau_{m,i} \in \mathcal{V}_m, \tau_{n,j} \in \mathcal{V}_n$, and $\tau_{m,i} \neq \tau_{n,j}$, we readily get the following inequalities.

$$T^{m,i}_{\text{finish}} \le (2 - \alpha_{n,j} - \alpha_{m,i}) \times \mathcal{Z} + T^{n,j}_{\text{start}} + \mathcal{Z} \times (1 - \delta_{i,j}) \tag{23}$$

$$T^{n,j}_{\text{finish}} \le (2 - \alpha_{n,j} - \alpha_{m,i}) \times \mathcal{Z} + T^{m,i}_{\text{start}} + \mathcal{Z} \times \delta_{i,j}. \tag{24}$$

Similarly, $\forall \tau_{m,i} \in \mathcal{V}_m, \tau_{n,j} \in \mathcal{V}_n, \tau_{m,i} \neq \tau_{n,j}, \pi \in [1, M], h \in [1, H]$, we can deduce

$$T^{m,i}_{\text{finish}} \le (2 - \beta_{n,j,\pi} - \beta_{m,i,\pi}) \times \mathcal{Z} + T^{n,j}_{\text{start}} + \mathcal{Z} \times (1 - \delta_{i,j}) \tag{25}$$

$$T^{n,j}_{\text{finish}} \le (2 - \beta_{n,j,\pi} - \beta_{m,i,\pi}) \times \mathcal{Z} + T^{m,i}_{\text{start}} + \mathcal{Z} \times \delta_{i,j} \tag{26}$$

$$T^{m,i}_{\text{finish}} \le (2 - \gamma_{n,j,h} - \gamma_{m,i,h}) \times \mathcal{Z} + T^{n,j}_{\text{start}} + \mathcal{Z} \times (1 - \delta_{i,j}) \tag{27}$$

$$T^{n,j}_{\text{finish}} \le (2 - \gamma_{n,j,h} - \gamma_{m,i,h}) \times \mathcal{Z} + T^{m,i}_{\text{start}} + \mathcal{Z} \times \delta_{i,j}. \tag{28}$$

B. Proposed Approach

As presented previously, our studied problem can be expressed as an ILP problem. Although existing ILP solvers (e.g., an open-source ILP solver in [23]) are capable of deriving an optimal solution, they may incur unaffordable time overheads for dealing with large-scale problem instances. Given this dilemma, we develop a reliability-driven heuristic task offloading approach to achieve the goal of energy consumption minimization. As shown in Fig. 1, our approach first exploits the metric of communication-computation-ratio (CCR) to quantify differentiated DAG energy demands. Therefore, the iterative self-organizing data analysis technique algorithm (ISODATA) is utilized to group DAG applications based on their CCR values. At this moment, an individual DAG cluster can be distinguished into a computation-intensive or communication-intensive category. Afterward, for computation-intensive DAG clusters, our approach leverages the incremental control technique to determine energy-efficient task execution modes based on an analysis of computation energy optimality. Meanwhile, for communication-intensive DAG clusters, our approach designs a DAG type conversion technique to reduce the intertask communication overheads. Thanks to this conversion operation, communication-intensive DAG clusters are now inverted to computation-intensive DAG clusters, and thus, their task

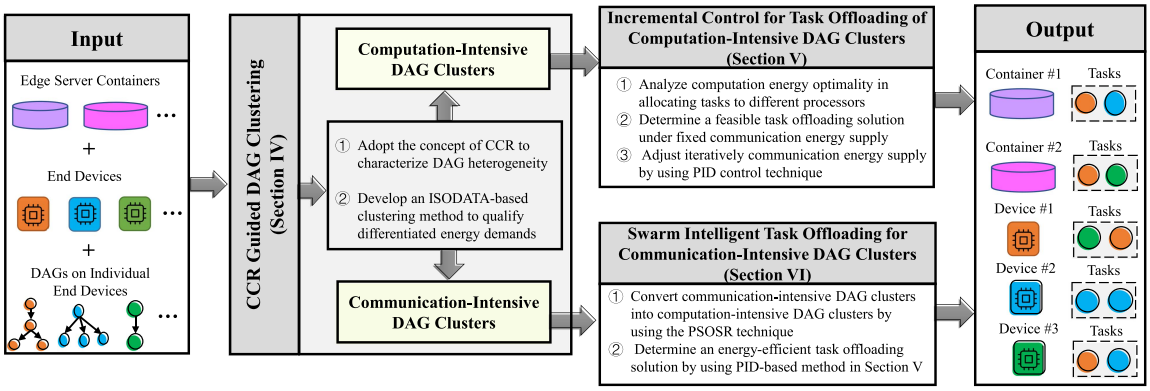

Fig. 1. Overview of our proposed approach.

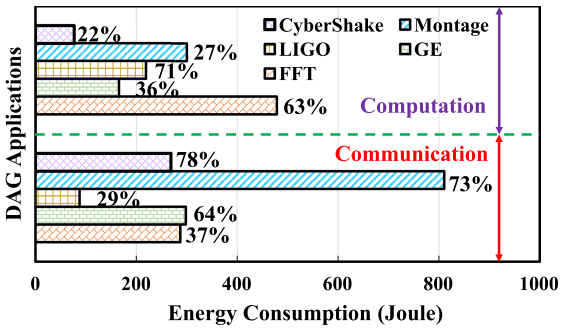

Fig. 2. Example of DAG energy dissipation composition.

offloading solution can be derived by invoking the incremental control-based method.

IV. CCR-Guided DAG Clustering

In this section, we first present an observation on DAG heterogeneity in energy demands, and then, exhibit our DAG clustering method followed by the DAG clustering algorithm.

A. Observation on DAG Heterogeneity in Energy Demands

As explored in [24], the heterogeneity of DAG applications can be depicted from multiple perspectives, such as the CCR, DAG size, parallelism factor, etc. Among these indicators, CCR is widely adopted to characterize the DAG application heterogeneity in computation and communication overheads. Specifically, the CCR value of a given DAG application is calculated as the average communication overhead divided by the average computation overhead during its execution on a specific hardware platform. Essentially, this metric implies either the communication energy or computation energy occupies a larger proportion of the whole energy dissipation of a DAG application. As an example, Fig. 2 demonstrates the energy dissipation composition of some representative real-life DAG applications, including fast Fourier transform (FFT), Gaussian elimination (GE), CyberShake, Montage, and LIGO inspiral analysis. As observed, the CCR values of different DAG applications vary significantly. For example, on one hand, the two applications of CyberShake and Montage maintain higher CCR values of 3.5 and 2.7, respectively. On the other hand, the remaining three applications of FFT, GE, and LIGO inspiral analysis attain lower CCR values of 0.6, 0.8, and 1.4, respectively. This observation inspires us to leverage the DAG CCR values to quantify differentiated energy demands in the communication and computation of DAG applications.

B. DAG Clustering Method

We have known that DAG applications have a great distinction in computation and communication energy demands. Inspired by this observation, it is natural to dedicate task offloading methods for the computation-intensive DAG applications and the communication-intensive DAG applications, respectively. We select the popular ISODATA technique to cluster DAGs into multiple groups. Compared with traditional clustering methods, ISODATA is adaptive to the cluster number by introducing novel merging and partitioning mechanisms [25]. During every round of grouping elements, the merging mechanism monitors both the size of each cluster and the distance between any two clusters. If the size of a single cluster is small enough, it will be wisely merged into an adjacent cluster. Meanwhile, if the distance between two clusters is less than a particular threshold, they will be combined as a new cluster. On the other hand, the partitioning mechanism keeps track of the number of elements and the average dispersion degree of all elements in an individual cluster. If the number of elements or the average dispersion degree of all elements is large enough, this cluster will be split into two small clusters immediately. We should emphasize that a valid distance criterion should be given in advance to measure the distance between merged elements ahead of running the ISODATA method. In the context of our DAG clustering, we adopt the metric of CCR value as a distance criterion based on the observation on DAG heterogeneity in energy demands, as detailed in the following section.

C. Algorithm of CCR-Guided DAG Clustering

We develop a CCR-guided DAG clustering scheme exploiting the ISODATA technique, as shown in Algorithm 1. Line 1 derives the CCR values of individual DAG applications. For application

Algorithm 1: CCR-Guided DAG Clustering Scheme.

Input: 1) $\{G_1, G_2, \cdots, G_M\}$, 2) $\{D_1, D_2, \cdots, D_M\}$, 3) a set of thresholds $\{U_{\text{limit}}^{(1)}, U_{\text{limit}}^{(2)}, U_{\text{limit}}^{(3)}, U_{\text{limit}}^{(4)}\}$.

1 Derive CCR values $\{\text{CCR}_1, \text{CCR}_2, \cdots, \text{CCR}_M\}$;
2 Initialize cluster number: $K \leftarrow K_0$;
3 Randomly choose K clustering centers $\{w_1, w_2, \cdots, w_K\}$;
4 **for** $m = 1$ to M **do**
5 $k \leftarrow \arg\min ||\text{CCR}_m - \text{CCR}_{w_k}||, \forall k = 1, 2, \cdots K$;
6 Append DAG application G_m to cluster Φ_{w_k};
7 **while** Termination condition is not met **do**
8 **for** $k = 1$ to K **do**
9 **if** $|\Phi_{w_k}| < U_{\text{limit}}^{(1)}$ **then**
10 **for** $i = 1$ to $|\Phi_{w_k}|$ **do**
11 Append $G_i \in \Phi_{w_k}$ to its nearest cluster and update the center of the nearest cluster;
12 Decrease cluster number: $K \leftarrow K - 1$;
13 **else**
14 **if** $|\Phi_{w_k}| > U_{\text{limit}}^{(2)}$ **then**
15 Split cluster Φ_{w_k} into two small clusters;
16 Increase cluster number: $K \leftarrow K + 1$;
17 **for** $k = 1$ to K **do**
18 **for** $j = 1$ to K **do**
19 Derive distance: $H_{k,j} \leftarrow ||\text{CCR}_{w_k} - \text{CCR}_{w_j}||$;
20 **if** $H_{k,j} < U_{\text{limit}}^{(3)}$ **then**
21 Merge Φ_{w_k} and Φ_{w_j} into a new cluster;
22 Decrease cluster number: $K \leftarrow K - 1$;
23 Calculate CCR variance V_{w_k} of elements in Φ_{w_k}
24 **if** $V_{w_k} > U_{\text{limit}}^{(4)}$ **then**
25 Split cluster Φ_{w_k} into two small clusters;
26 Increase cluster number: $K \leftarrow K + 1$;
27 **return** A set of DAG clusters $\Phi = \{\Phi_1, \Phi_2, \cdots, \Phi_K\}$.

G_m, its CCR value CCR_m is estimated by

$$\text{CCR}_m = \sum_{i=1}^{\xi_m} \left(a_1 \times W_{m,i} + a_2 \times \sum_{\tau_{m,p} \in \mathcal{I}_{\text{child}}^{m,i}} \eta_{m,i,p} \right) \quad (29)$$

where a_1 and a_2 are the energy dissipation coefficients in respect of computation and communication of the target system, respectively. Line 2 initializes the total number K of clusters. Line 3 selects K clustering centers $\{w_1, w_2, \ldots, w_{K_d}\}$ in a random manner. Lines 4–6 iteratively allocate each application G_m to cluster Φ_{w_k} with a smallest CCR difference between CCR_m and CCR_{w_k}. After the aforementioned initialization operations, our algorithm then enters a loop of finding an optimal clustering solution in lines 7–26. Specifically, line 9 checks whether or not the total number of elements in cluster Φ_{w_k} is below a predefined lower boundary. If yes, all elements in cluster Φ_{w_k} are assigned to an adjacent cluster and the overall number K_d of clusters is decreased (lines 10–12). On the contrary, if the total number of elements in cluster Φ_{w_k} is greater than a predefined upper boundary, cluster Φ_{w_k} will be partitioned into two small clusters and the overall number K of clusters is accordingly increased (lines 13–16). Lines 17–20 ascertain whether the distance between any two clusters is closer enough. If yes, lines 21–22 first merge two similar clusters into a new cluster, and then, update the cluster number K. Meanwhile, if the CCR variance of all elements in cluster Φ_{w_k} exceeds an allowable threshold, cluster Φ_{w_k} will be split into two small clusters in lines 23–26. The entire algorithm ends up outputting the final clustering result Φ in line 27.

Algorithm 2: Incremental Control Algorithm for Task Offloading of Computation-Intensive DAG Clusters.

Input: Task set Ω_{w_k} of a computation-intensive cluster.

1 Sort tasks in Ω_{w_k} in the descending order of task power coefficient;
2 Sort processors in $\Theta = \{\Theta_1, \Theta_2, \cdots, \Theta_{|\Phi_{w_k}|}\}$ in the ascending order of processor power coefficient;
3 **while** Termination condition is not met **do**
4 Set PID controller for communication energy restriction;
5 **for** $i = 1$ to ϕ **do**
6 **for** $h = 1$ to H **do**
7 $\Delta_{h,i} \leftarrow \text{CheckConstraints}(\tau_i, C_h)$;
8 **if** $\Delta_{h,i} ==$ true **then**
9 Assign task τ_i to container C_h;
10 Remove task τ_i: $\Omega_{w_k} \leftarrow \Omega_{w_k} - \tau_i$;
11 **break**;
12 **if** τ_i is in task set Ω_{w_k} **then**
13 **for** $m = 1$ to $|\Phi_{w_k}|$ **do**
14 $\Delta_{m,i} \leftarrow \text{CheckConstraints}(\tau_i, D_m)$;
15 **if** $\Delta_{m,i} ==$ true **then**
16 Allocate task τ_i to end device D_m;
17 **break**;
18 Update task offloading solution $\mathcal{O}_{w_k}$;
19 **return** Task offloading solution $\mathcal{O}_{w_k}$.

V. Incremental Control for Task Offloading of Computation-Intensive DAG Clusters

In this section, we design an incremental control scheme for the task offloading of computation-intensive DAG clusters.

A. Observation on Computation Energy Optimality

Algorithm 1 produces a collection of DAG groups by leveraging the ISODATA clustering technique. Afterward, we can classify each DAG cluster into a computation-intensive category or a communication-intensive category based on its resulting CCR value on average, that is,

$$\text{CCR}(\Phi_{w_k}) = \frac{1}{|\Phi_{w_k}|} \sum\nolimits_{G_m \in \Phi_{w_k}} \text{CCR}_m. \quad (30)$$

If $\text{CCR}(\Phi_{w_k})$ is below a threshold, the DAG cluster Φ_{w_k} is naturally classified into the computation-intensive category; otherwise, it is, therefore, classified into the communication-intensive category. Suppose that Φ_{w_k} is a computation-intensive DAG cluster and $\Omega_{w_k} = \{\tau_1, \tau_2, \ldots, \tau_\phi\}$ is a collection of all tasks in DAG cluster Φ_{w_k}. We neglect the computation energy analysis of those tasks selecting the remote execution mode since they only incur communication energy consumption of end devices. Let $D(\Phi_{w_k}) = \{D_1, D_2, \ldots, D_{|\Phi_{w_k}|}\}$ be a group of end devices whose tasks are in DAG cluster Φ_{w_k}. Then, let $\Omega'_{w_k} = \{\tau_1, \tau_2, \ldots, \tau_{\phi'}\}$ be a set of the tasks selecting either the local or D2D execution mode. Let τ_ε $(1 \leq \varepsilon \leq \phi')$ denote the εth element in task set Ω'_{w_k}, and the index of the end device owning task τ_ε is omitted for easy presentation. According to

(5), the computation energy of task subset Ω'_{w_k} is derived by

$$E(\Omega'_{w_k}) = \sum_{m=1}^{|\Phi_{w_k}|} \sum_{\tau_\varepsilon \in \Omega'_m} \psi_m \times v_m^2 \times \mu_\varepsilon \times W_\varepsilon \quad (31)$$

where $\Omega'_m \subseteq \Omega'_{w_k}$ represents a group of real-time tasks that are locally executed or offloaded to be completed on end device D_m. Furthermore, let $\theta = [\theta_1, \theta_2, \ldots, \theta_{|\Phi_{w_k}|}]$ represent a vector storing processor-dependent parameters, where $\theta_m = \psi_m \times v_m^2$ refers to as the power coefficient of processor Θ_m. Similarly, let $\mathcal{Y} = [\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_{|\Phi_{w_k}|}]$ represent a vector storing task-dependent parameters, where $\mathcal{Y}_m = \sum_{\tau_\varepsilon \in \Omega'_m} (\mu_\varepsilon \times W_\varepsilon)$ refers to as the power coefficient of task subset Ω'_m. Obviously, $E(\Omega'_{w_k})$ can be rewritten as the product of processor-dependent parameters and task-dependent parameters, i.e., $E(\Omega'_{w_k}) = \theta_1 \times \mathcal{Y}_1 + \theta_2 \times \mathcal{Y}_2 + \cdots + \theta_{|\Phi_{w_k}|} \times \mathcal{Y}_{|\Phi_{w_k}|}$. At this moment, we observe the optimality of computation energy $E(\Omega'_{w_k})$, shown as follows.

Theorem 1: For a given task set $\Omega'_{w_k} = \{\tau_1, \tau_2, \ldots, \tau_{\phi'}\}$, when the computation energy consumption $E(\Omega'_{w_k})$ is minimized under $\theta_1 \le \theta_2 \cdots \le \theta_{|\Phi_{w_k}|}$, then the inequality $\mathcal{Y}_1 \ge \mathcal{Y}_2 \ge \cdots \ge \mathcal{Y}_{|\Phi_{w_k}|}$ holds.

Proof: Please refer to the Appendix. □

B. Incremental Control Algorithm

We learn from Theorem 1 that an optimal computation energy consumption is achieved when the processors with smaller power coefficients are to execute the tasks with larger power coefficients. Inspired by this observation, we put forward an incremental control scheme in Algorithm 2 for the task offloading of computation-intensive DAG clusters. Line 1 sorts all tasks in set Ω_{w_k} in the descending order of task power coefficient while maintaining the topological order. Line 2 sorts all processors in set Θ in the order of processor power coefficient from low to high. Lines 3–18 enter into a procedure of iteratively searching for an optimal task offloading solution with the help of proportional-integral-derivative (PID) control techniques. In the procedure, line 3 judges whether or not the termination condition is met. If no, line 4 sets the PID controller for communication energy restriction, i.e., none of the communication energy $E^{m,i,p}_{\text{localcom}}$, $E^{m,i,\pi}_{\text{d2dcom}}$, and $E^{m,i,h}_{\text{remcom}}$ can exceed threshold $E^{\text{limit}}_{\text{comm}}$. In this step, $E^{\text{limit}}_{\text{comm}}$ is updated by [13]

$$\begin{aligned} E^{\text{limit}}_{\text{comm}}(u+1) = {} & E^{\text{limit}}_{\text{comm}}(u) - Q_1 \times R_{\text{ratio}}(u) - Q_2 \\ & \times \sum_{\ell=1}^{U_1} R_{\text{ratio}}(u - U_1 + \ell) - Q_3 \\ & \times \frac{R_{\text{ratio}}(u) - R_{\text{ratio}}(u - U_2)}{U_2} \end{aligned} \quad (32)$$

where Q_1, Q_2, and Q_3 denote the proportional, integral, and derivative coefficients of the PID controller, respectively. $R_{\text{ratio}}(u)$ refers to the difference between the desired reliability satisfaction ratio (i.e., 100%) and the number of tasks meeting reliability constraints divided by overall task number at the uth iteration. U_1 denotes the number of iterations during which the integral errors are accumulated. U_2 represents the number of iterations during which the derivative errors are measured.

Algorithm 3: Swarm Intelligent Task Offloading Algorithm for Communication-Intensive DAG Clusters.

Input: Task set Γ_{w_d} of a communication-intensive cluster.
1 Produce initial feasible particles $\mathcal{X} = \{S_1, S_2, \cdots, S_J\}$;
2 Evaluate fitness values of all initial particles in $\mathcal{X}$;
3 **while** Termination condition is not met **do**
4 Sort particles in the descending order of fitness values;
5 **for** $j = 1$ to J **do**
6 Select l_j binary variables to relax value ranges;
7 $S^{j,1}_{\text{child}} \leftarrow \text{OffspringContinuous}(S_{j,1})$;
8 $S^{j,2}_{\text{child}} \leftarrow \text{OffspringDiscrete}(S_{j,2})$;
9 Generate a complete offspring: $S^{j}_{\text{child}} \leftarrow S^{j,1}_{\text{child}} + S^{j,2}_{\text{child}}$;
10 Round continuous variables: $S^{j}_{\text{child}} \leftarrow \text{SR}(S^{j}_{\text{child}})$;
11 Check CCR value: $\Delta_j \leftarrow \text{CheckCCR}(S^{j}_{\text{child}})$;
12 **if** $\Delta_j ==$ ture **then**
13 **if** $\text{Compare}(S^{j}_{\text{child}}, S_j)$ **then**
14 Update particle S_j: $S_j \leftarrow S^{j}_{\text{child}}$;
15 Adjust evolutionary parameters for particle S_j;
16 Select best particle: $S_{\text{opt}} \leftarrow \text{Select}(\mathcal{X})$;
17 Calculate task power coefficients : $\mathcal{V}_\Gamma \leftarrow \text{PowerFactor}(\Gamma_{w_d}, S_{\text{opt}})$;
18 Call Algorithm 2 to produce offloading solution $\mathcal{O}_{w_d}$;
19 **return** Task offloading solution $\mathcal{O}_{w_d}$.

After PID controller settings, lines 6–12 aim to select a suitable container C_h as the computation offloading destination for every task τ_i. Here, CheckConstraints(τ_i, C_h) is a test function to check whether or not all constraints in (15)–(28) are satisfied if task τ_i is to execute on container C_h. If flag $\Delta_{h,i}$ is true, lines 9–10 first allocate task τ_i to container C_h, and then, accordingly delete task τ_i from task set Ω_{w_k}. If task τ_i is rejected by all containers, lines 12–17 adopt the first-fit manner to determine an energy-efficient end device for task execution based on Theorem 1. Line 18 updates the task offloading solution $\mathcal{O}_{w_k}$ obtained by the current iteration. The whole algorithm is terminated with outputting task offloading solution $\mathcal{O}_{w_k}$ for task set Ω_{w_k} in line 19. We should point out that Algorithm 2 can be run in parallel for each DAG cluster, thereby significantly reducing the time overhead of task offloading solution generation.

VI. Swarm Intelligent Task Offloading for Communication-Intensive DAG Clusters

In this section, we design a swarm intelligent task offloading scheme for communication-intensive DAG clusters.

A. DAG Type Transformation

We devise a novel DAG type transformation method, named PSOSR, to convert communication-intensive DAG clusters into computation-intensive DAG clusters. This method is inspired by the idea of hybrid algorithm design explored in [26] and [27], and it combines the advantages of the state-of-art version of particle swarm optimization (PSO) [28] and the sequential rounding (SR) [29] in effectively solving complex problems. Let $\Gamma_{w_d} = \{\tau_1, \tau_2, \ldots, \tau_L\}$ denote a set of all tasks in a communication-intensive DAG cluster $\Phi_{w_d} \in \Phi$, where the index of the end device owning task τ_l $(1 \le l \le L)$ is omitted for easy presentation. Further, let $X_{w_d} = \{X_1, X_2, \ldots, X_L\}$ be a set of binary merging points, where variable X_l $(1 \le l \le L)$ is set to 1 only

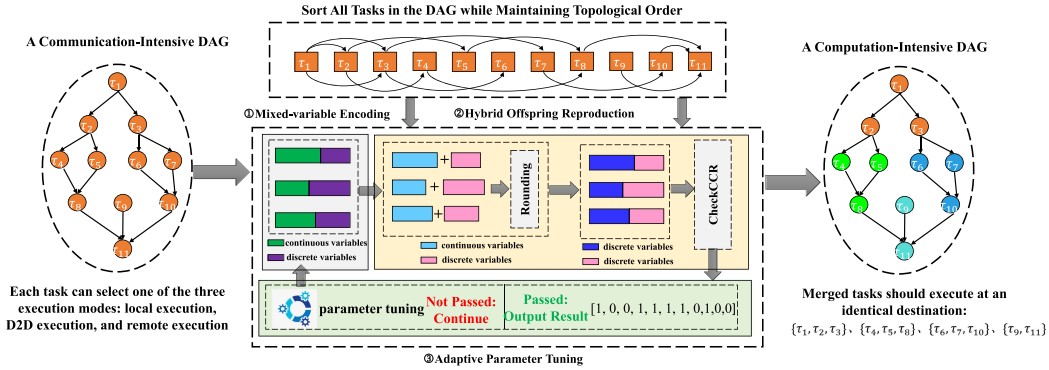

Fig. 3. Example of our DAG type transformation method.

if task τ_l is selected as a merging point. To reduce intertask communication overheads, we restrict that task τ_l and all its direct successors not selected as merging points need to execute at an identical destination.

Specifically, the PSOSR method relaxes l merging points to take arbitrary real numbers from [0, 1], i.e., the inequality $0 \leq X_1, X_2, \ldots, X_l \leq 1$ holds. At this moment, there obviously coexist continuous variables $\{X_1, X_2, \ldots, X_l\}$ and discrete variables $\{X_{l+1}, X_{l+2}, \ldots, X_L\}$. Afterward, the up-to-date PSO variant detailed in [28] is adopted to address the mixed-variable task merging problem. Unlike other PSO variants, this latest PSO technique is characterized by three stages of mixed-variable encoding, hybrid offspring reproduction, and adaptive parameter tuning. In the first stage, the particle position vector (i.e., transformation solution) is divided into two varied segments for encoding continuous and discrete variables, respectively. Consequently, different evolutionary operators can be exploited to separately evolve continuous and discrete variables. In the second stage, two reproduction schemes are designed to create a fraction of the offspring position vector linked with continuous and discrete variables in parallel. Then, a complete offspring particle is readily constructed by putting together the two position vector segments. To exactly determine merging points, we incorporate the SR technique [29] to round continuous variables to either 0 or 1 via comparing with a predefined threshold. Following the aforementioned stages, the third stage concentrates on optimizing critical evolutionary parameters for the next iteration. Fig. 3 depicts an example of our DAG type transformation method assuming that the current DAG cluster only contains one element.

B. Swarm Intelligent Task Offloading Algorithm

Algorithm 3 presents our swarm intelligent task offloading scheme for communication-intensive DAG clusters. Line 1 produces initial feasible particles $\mathcal{X} = \{S_1, S_2, \ldots, S_J\}$. Line 2 evaluates every initial feasible particle in terms of fitness, which is calculated as the difference between the CCR value of the particle and the predefined CCR threshold. Lines 3–18 enter into an iterative procedure of finding task offloading solutions. Specifically, if the termination condition is not met, line 4 sorts all particles in a descending order. For every particle S_j, line 6 randomly selects a total of l_j binary variables to relax their value ranges. At this step, particle S_j is divided into two segments $S_{j,1}$ and $S_{j,2}$ of separately storing continuous and discrete variables. Afterward, line 7 builds an offspring $S_{\text{child}}^{j,1}$ for continuous variables in segment $S_{j,1}$ by invoking function OffspringContinuous$(S_{j,1})$. Likewise, line 8 produces an offspring $S_{\text{child}}^{j,2}$ for discrete variables in segment $S_{j,2}$ by calling function OffspringDiscrete$(S_{j,2})$. Line 9, therefore, generates a complete offspring S_{child}^{j} by combining $S_{\text{child}}^{j,1}$ and $S_{\text{child}}^{j,2}$. Line 10 rounds continuous variables in segment $S_{j,1}$ by using SR function SR(S_{child}^{j}). Line 11 checks whether or not the current CCR value of task set Γ is below a predefined CCR threshold when task merging solution S_{child}^{j} is applied to task set Γ. If yes, flag Δ_j is set to true. Line 13 compares offspring S_{child}^{j} and parent S_j by exploiting comparison function Compare$(S_{\text{child}}^{j}, S_j)$. If offspring S_{child}^{j} is superior to parent S_j, lines 14–15 replace parent S_j with offspring S_{child}^{j} and adjust evolutionary parameters of particle S_j, respectively. After all particles are examined, line 16 selects a particle S_{opt} with best fitness from swam $\mathcal{X}$ by using selection function Select$(\mathcal{X})$. Accordingly, line 17 derives task power coefficients by using function PowerFactor(Γ, S_{opt}). In this step, the task and all its direct successors not selected as merging points are constructed as a big task $\tau_l^{\star}$, and their communication energy overheads are equal to zero [see (6)]. Since the DAG type transformation has been finished right now, line 17 calls Algorithm 2 to produce offloading solution $\mathcal{O}_{w_d}$ for task set Γ_{w_d}. When the termination condition is satisfied, line 19 exists after outputting solution $\mathcal{O}_{w_d}$. Similar to Algorithm 2, Algorithm 3 can also be performed in parallel for communication-intensive DAG clusters, thereby saving the time overheads in producing an energy-efficient task offloading solution.

VII. Simulation

In this section, we conduct simulation experiments to validate the effectiveness of our solution for synthetic applications.

A. Simulation Settings

In simulation settings, the operating frequency f_m of the processor Θ_m is randomly selected from $[500, 1500]$ MHz [9]. The Dell PowerEdge R930 server [30] equipped with an Intel Xeon E7-8894 24-core processor is selected as the edge server.

We suppose that a total of 15 containers are installed on our edge server and the operating frequency of one container is randomly selected from $[2000, 4500]$ MHz. The average fault arrival rates of individual processors and containers are both assumed to be in $[4 \times 10^{-6}, 7 \times 10^{-5}]$ according to their computation capacities [31], [32]. The bit error rate of one communication link is chosen from $[2 \times 10^{-6}, 5 \times 10^{-5}]$ [13]. The D2D transmission and receiving power of a single end device falls into $[200, 1000]$ and $[100, 800]$ mW, respectively [5]. The D2D communication bandwidth between any two end devices varies from 20 to 100 MHz [5]. The transmission power of an arbitrary end device in the remote execution mode is selected from $[600, 1500]$ mW, and its communication bandwidth to the destination container is picked from [100, 400] MHz. We leverage the tool TGFF [33] to generate diverse synthetic DAG applications with the task number varied in [20,300]. Accordingly, a set of heterogeneous DAG applications with varied CCR values is readily produced. The reliability goal of each DAG application is randomly chosen from [0.70,0.9999]. The proportional, integral, and derivative coefficients of our PID controller are set to 0.5, 0.005, and 0.1, respectively [13].

As summarized in Table I, we conduct energy optimization for large-scale CPS with joint considerations of task latency, task reliability, task dependence, and end–end–edge collaboration. We pick the following representative benchmarking strategies that have the most similar concerns to our problem.

1) *ELYO* [5] exploits the Lyapunov optimization technique to achieve time-average energy dissipation minimization for independent tasks. It takes both task local and D2D execution modes into account but ignores the task remote execution mode provided by edge servers.
2) *ELDM* [7] leverages the Lagrange duality method to decide one of the three execution modes for every task, thereby minimizing the system overall energy consumption. However, neither the intertask dependencies nor the task reliability demands are considered.
3) *ELGT* [9] is a game theoretic task offloading algorithm to jointly minimize the energy optimization and processing latency of independent tasks with the help of end–end–edge collaboration. However, it fails to consider both the toleration of soft errors during task executions and the occurrence of bit errors during intertask communication.
4) *ERPS* [15] aims at minimizing the energy consumption of dependent real-time tasks by using processor-merging and slack time reclamation methods. It considers the toleration of soft errors during task executions but neglects the occurrence of bit errors during intertask communication. Moreover, the computation paradigm of end–end–edge collaboration is not used during energy optimization.
5) *EILP* utilizes an open-source ILP solver [23] to tackle the ILP problem of energy minimization formulated in Section III-A. As mentioned earlier, this scheme yields a globally optimal task offloading solution, but is highly likely to incur huge runtime overheads.

B. Simulation Results

In the comparative study, we conduct a total of 100 experiments to obtain averaged evaluation data. Table II exhibits the energy dissipation and the corresponding energy savings achieved by our approach when running synthetic DAG applications. On one hand, we observe that our approach significantly reduces the whole energy consumption of end devices, especially when more end devices are involved in the task offloading process. On the other hand, we also see that our approach is inferior to benchmarking algorithm EILP, with 9.86% degradation on average in terms of energy consumption.

Table III lists the runtime overheads of task offloading algorithms and the resultant speedup attained by our approach when running synthetic DAG applications. The results in this table confirm the effectiveness of our approach in shortening the runtime spent on deriving desirable task offloading solutions. In addition, the results also reflect that the runtime overheads of our approach tend to grow slowly rather than rapidly as the number of end devices increases. This is mainly because the incorporation of ISODATA technique into our approach empowers the parallel searching for task offloading solutions of different DAG clusters.

We further explore the schedulability of six task offloading algorithms. The schedulability of an algorithm is the ratio of the number of DAG application instances satisfying specific constraints to the total number of DAG application instances under test (i.e., 100 in our experiments). Table IV shows the results when only considering the task dependence or deadline constraints of synthetic DAG applications. As observed, when only considering the intertask dependence constraints, the schedulability of benchmarking algorithms ELYO, ELGT, and ELDM is merely 43%, 59%, and 42% on average, respectively. When only considering the task deadline constraints, the schedulability of benchmarking algorithms ELYO, ELGT, ELDM, and ERPS is 38%, 53%, 36%, and 65% on average, respectively. This is because the four benchmarking algorithms are customized for the latency-aware task offloading of a single application while neglecting the concurrent task offloading of individual applications with distinct timing requirements. Conversely, our approach and benchmarking algorithm EILP always achieve 100% schedulability when imposed either the intertask dependence or task deadline constraints.

Similarly, Table V presents the schedulability of six task offloading algorithms when considering the task reliability and all constraints of synthetic DAG applications. As observed, when benchmarking algorithms ELYO, ELGT, and ELDM are all imposed on the task reliability constraints, they inevitably suffer from undesirable reliability violations (up to 44%) due to the neglect of tolerating both bit errors and soft errors. Although benchmarking algorithm ERPS can handle the soft errors during task executions, it ignores the tolerance of bit errors during intertask communication, thereby resulting in a lower task overall reliability. Further, when considering all design constraints (i.e., intertask dependence, task deadline, and task reliability constraints), benchmarking algorithms ELYO, ELGT, ELDM, and ERPS cannot realize 100% schedulability. In contrast, our approach and benchmarking algorithm EILP

TABLE II
ENERGY DISSIPATION ACHIEVED BY TASK OFFLOADING ALGORITHMS WHEN RUNNING SYNTHETIC APPLICATIONS

The Number of end devices	Energy dissipation (Joule) of task offloading solutions						Energy savings (%) of our approach				
	Proposed	ELYO	ELGT	ELDM	ERPS	EILP	ELYO	ELGT	ELDM	ERPS	EILP
M=100	1853.57	2020.68	2003.23	1967.78	2405.74	1810.65	8.27	7.47	5.80	22.95	-2.37
M=150	2430.20	2683.82	2672.10	2616.29	3108.45	2341.68	9.45	9.05	7.11	21.82	-3.78
M=200	3303.82	3717.17	3673.25	3558.59	4686.14	3159.13	11.12	10.06	7.16	29.50	-4.58
M=250	4450.09	5159.52	5127.21	4857.88	6325.47	4235.76	13.75	13.21	8.39	29.65	-5.06
M=300	5789.71	6802.62	6792.84	6504.69	9365.12	5483.72	14.89	14.77	10.99	38.18	-5.58
M=350	7762.89	9194.46	9237.55	8760.81	11588.18	7311.75	15.57	15.96	11.39	33.01	-6.17
M=400	10171.62	12265.30	12199.01	11630.95	16523.82	9477.84	17.07	16.62	12.55	38.44	-7.32
M=450	13680.71	16875.19	16522.56	15975.68	22793.51	12501.57	18.93	17.20	14.37	39.98	-9.43
M=500	18006.01	22726.26	22065.90	21261.57	29293.79	16308.32	20.77	18.40	15.31	38.53	-10.41
M=550	23944.70	31328.93	29639.60	28948.67	44339.34	21599.05	23.57	19.21	17.29	46.00	-10.86
M=600	27129.92	37374.18	36986.66	35644.44	60504.79	24390.39	27.41	26.65	23.89	55.16	-11.23
M=650	29874.84	45430.11	41791.22	41471.74	111161.00	26633.54	34.24	28.51	27.96	73.12	-12.17
M=700	33612.12	56132.46	49445.69	47528.34	104177.95	29750.50	40.12	32.02	29.28	67.74	-12.98
M=750	37983.90	71952.83	64106.89	57582.63	191660.18	33533.95	47.21	40.75	34.04	80.18	-13.27
M=800	41977.17	84427.13	74119.10	68186.69	263526.59	36958.24	50.28	43.37	38.44	84.07	-13.58
M=850	47636.31	109383.04	90705.20	83240.79	325394.92	41734.99	56.45	47.48	42.77	85.36	-14.14
M=900	53273.68	136214.99	106968.68	99349.99	238212.14	46645.38	60.89	50.20	46.38	77.64	-14.21
M=950	58979.04	160399.90	136673.09	116117.09	1444067.32	51299.51	63.23	56.85	49.21	95.92	-14.97
M=1000	66245.86	303184.72	165735.39	155007.39	711562.90	57445.25	78.15	60.03	57.26	90.69	-15.32
Avg.	25689.80	58803.86	46129.75	42642.74	189510.39	22769.54	32.18	27.78	24.19	55.15	-9.86

TABLE III
RUNTIME OVERHEADS OF TASK OFFLOADING ALGORITHMS WHEN RUNNING SYNTHETIC APPLICATIONS

The Number of end devices	Measured runtime (seconds) of task offloading algorithms						Speedup (times) of our approach against peers				
	Proposed	ELYO	ELGT	ELDM	ERPS	EILP	ELYO	ELGT	ELDM	ERPS	EILP
M=100	3.67	34.62	61.15	50.37	90.47	157.37	8.44	15.67	12.73	23.66	41.90
M=150	4.39	52.39	69.20	73.15	95.27	227.85	10.94	14.78	15.68	20.72	50.94
M=200	5.28	70.84	91.64	98.50	131.97	356.70	12.43	16.37	17.67	24.01	66.60
M=250	6.18	108.82	111.31	97.22	169.01	424.58	16.59	17.00	14.72	26.33	67.65
M=300	6.90	127.55	141.74	132.45	204.95	584.56	17.49	19.55	18.20	28.71	83.73
M=350	7.38	163.55	167.96	175.52	228.28	659.84	21.17	21.77	22.80	29.95	88.45
M=400	7.96	206.61	190.70	196.93	239.59	810.37	24.94	22.94	23.73	29.08	100.75
M=450	8.36	225.30	216.86	229.88	287.92	996.63	25.95	24.94	26.50	33.44	118.21
M=500	8.97	244.69	275.03	271.61	357.19	1163.49	26.26	29.65	29.26	38.80	128.64
M=550	9.14	276.00	314.72	302.26	332.01	1183.41	29.20	33.43	32.07	35.33	128.48
M=600	9.67	301.86	365.77	352.63	441.29	1381.71	30.21	36.81	35.45	44.62	141.84
M=650	10.02	350.79	420.46	397.52	416.34	1652.02	34.02	40.98	38.69	40.57	163.94
M=700	10.38	374.40	448.65	466.19	499.74	1749.47	35.06	42.21	43.90	47.13	167.48
M=750	11.08	473.90	589.44	528.34	563.16	1942.78	41.77	52.20	46.69	49.83	174.35
M=800	11.46	486.61	605.92	584.74	569.01	2380.57	41.46	51.87	50.03	48.65	206.73
M=850	12.44	544.68	738.81	664.25	670.09	2538.92	42.80	58.41	52.41	52.88	203.16
M=900	14.11	701.55	889.88	848.73	716.31	3219.28	48.72	62.07	59.16	49.77	227.18
M=950	14.80	806.73	1001.56	1018.90	937.86	3445.65	53.52	66.69	67.86	62.38	231.86
M=1000	15.82	972.40	1174.24	1284.92	1183.72	3994.27	60.45	73.21	80.20	73.80	251.42
Avg.	9.37	343.33	414.48	409.16	428.11	1519.45	30.60	36.87	36.20	39.98	139.12

TABLE IV
SCHEDULABILITY WHEN ONLY CONSIDERING TASK DEPENDENCE/DEADLINE CONSTRAINTS OF SYNTHETIC APPLICATIONS

The Number of end devices	Schedulability when only considering inter- task dependence constraints						Schedulability when only considering task deadline constraints					
	Proposed	ELYO	ELGT	ELDM	ERPS	EILP	Proposed	ELYO	ELGT	ELDM	ERPS	EILP
M=100	100%	59%	76%	61%	100%	100%	100%	55%	76%	57%	86%	100%
M=150	100%	56%	74%	58%	100%	100%	100%	51%	71%	53%	84%	100%
M=200	100%	54%	74%	50%	100%	100%	100%	47%	66%	48%	82%	100%
M=250	100%	51%	73%	50%	100%	100%	100%	46%	66%	47%	82%	100%
M=300	100%	51%	71%	49%	100%	100%	100%	45%	62%	44%	80%	100%
M=350	100%	49%	70%	49%	100%	100%	100%	44%	62%	42%	78%	100%
M=400	100%	48%	69%	48%	100%	100%	100%	44%	59%	41%	76%	100%
M=450	100%	46%	69%	47%	100%	100%	100%	44%	58%	41%	75%	100%
M=500	100%	45%	62%	42%	100%	100%	100%	38%	57%	40%	67%	100%
M=550	100%	43%	61%	40%	100%	100%	100%	36%	54%	39%	66%	100%
M=600	100%	42%	58%	39%	100%	100%	100%	36%	50%	34%	63%	100%
M=650	100%	41%	54%	38%	100%	100%	100%	35%	49%	32%	58%	100%
M=700	100%	41%	54%	38%	100%	100%	100%	32%	47%	31%	58%	100%
M=750	100%	35%	52%	36%	100%	100%	100%	32%	44%	30%	56%	100%
M=800	100%	35%	48%	32%	100%	100%	100%	30%	44%	29%	50%	100%
M=850	100%	35%	48%	29%	100%	100%	100%	28%	40%	28%	49%	100%
M=900	100%	31%	40%	28%	100%	100%	100%	23%	39%	21%	44%	100%
M=950	100%	28%	37%	27%	100%	100%	100%	23%	34%	19%	43%	100%
M=1000	100%	24%	28%	27%	100%	100%	100%	22%	33%	15%	39%	100%
Avg.	100%	43%	59%	42%	100%	100%	100%	38%	53%	36%	65%	100%

TABLE V
SCHEDULABILITY WHEN CONSIDERING TASK RELIABILITY AND ALL CONSTRAINTS OF SYNTHETIC APPLICATIONS

The Number of end devices	Schedulability when only considering task reliability constraints						Schedulability when only considering all design constraints					
	Proposed	ELYO	ELGT	ELDM	ERPS	EILP	Proposed	ELYO	ELGT	ELDM	ERPS	EILP
M=100	100%	75%	73%	73%	95%	100%	100%	50%	68%	50%	82%	100%
M=150	100%	74%	73%	72%	92%	100%	100%	49%	66%	46%	81%	100%
M=200	100%	72%	72%	71%	90%	100%	100%	44%	60%	42%	78%	100%
M=250	100%	70%	71%	69%	87%	100%	100%	42%	60%	41%	76%	100%
M=300	100%	70%	67%	69%	84%	100%	100%	39%	59%	39%	75%	100%
M=350	100%	70%	67%	66%	83%	100%	100%	39%	57%	38%	74%	100%
M=400	100%	65%	65%	65%	82%	100%	100%	38%	55%	38%	72%	100%
M=450	100%	59%	62%	62%	79%	100%	100%	37%	52%	37%	67%	100%
M=500	100%	58%	61%	61%	78%	100%	100%	35%	51%	37%	63%	100%
M=550	100%	58%	58%	61%	66%	100%	100%	33%	50%	36%	60%	100%
M=600	100%	56%	58%	60%	64%	100%	100%	30%	45%	28%	59%	100%
M=650	100%	54%	55%	55%	60%	100%	100%	30%	44%	27%	55%	100%
M=700	100%	52%	51%	54%	58%	100%	100%	27%	40%	27%	53%	100%
M=750	100%	46%	49%	54%	58%	100%	100%	27%	39%	26%	50%	100%
M=800	100%	44%	45%	48%	58%	100%	100%	24%	39%	26%	49%	100%
M=850	100%	42%	44%	47%	53%	100%	100%	22%	34%	21%	48%	100%
M=900	100%	40%	38%	47%	52%	100%	100%	21%	33%	17%	42%	100%
M=950	100%	32%	34%	47%	52%	100%	100%	18%	29%	15%	41%	100%
M=1000	100%	30%	33%	42%	51%	100%	100%	16%	24%	10%	37%	100%
Avg.	100%	56%	57%	59%	71%	100%	100%	33%	48%	32%	61%	100%

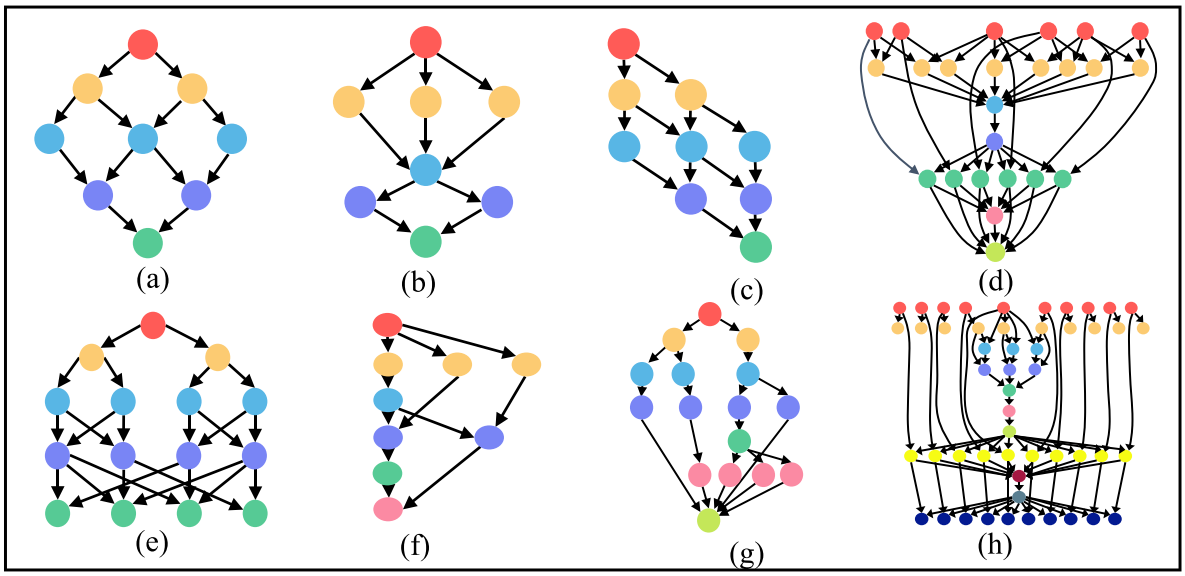

Fig. 4. Structure of partial real-life DAG applications under test.

always maintain 100% schedulability under varied settings of the number of end devices. However, compared with EILP, our approach can reduce the algorithm runtime overheads by a factor of 139.12 on average while achieving striking energy savings of end devices, as shown in Tables II and III.

VIII. FURTHER INVESTIGATION

We further investigate the effectiveness of our approach when running real-life DAG applications in CPS environments.

A. Real-Life Application Description

A collection of real-life DAG applications drawn from [24], [34], [35], [36], [37], and [38] are tested in this section, including the FFT, GE, mpegplay, madplay, tmndec, toast, Montage, CyberShake, Epigenomics, LIGO inspiral analysis, molecular dynamics code, Sipht, in-tree, out-tree, mean value analysis, Laplace equation solver, fork-join, LU-decomposition, face recognition, AIRSN, Chimera, navigator, SignalGuru, and TwitterSentiment. These benchmarks cover a wide spectrum of CPS applications, and hence, facilitate a comprehensive investigation on our approach and benchmarking algorithms. Fig. 4 illustrates the structures of partial DAG applications, while the structures of other DAG applications can be found in their original study. For every topological layer in a single DAG application, the number of tasks it contains can fluctuate to accommodate actual requirements. Hence, for each type of individual DAG applications, we randomly select the overall task number from $[50, 500]$ such that a plentiful of DAG application variants with varied CCR values are, hence, constructed. Similar to synthetic DAG applications, the reliability goal of each real-life DAG application is also randomly chosen from [0.70,0.9999].

TABLE VI
ENERGY DISSIPATION ACHIEVED BY TASK OFFLOADING ALGORITHMS WHEN RUNNING REAL-LIFE APPLICATIONS

The Number of end devices	Energy dissipation (Joule) of task offloading solutions						Energy savings (%) of our approach				
	Proposed	ELYO	ELGT	ELDM	ERPS	EILP	ELYO	ELGT	ELDM	ERPS	EILP
M=100	1893.82	2133.89	2018.80	1974.52	2471.28	1854.24	11.25	6.19	4.09	23.37	-2.13
M=150	2482.85	2841.77	2648.19	2611.94	3293.46	2419.43	12.63	6.24	4.94	24.61	-2.62
M=200	3366.62	3897.09	3655.50	3592.17	4502.47	3264.22	13.61	7.90	6.28	25.23	-3.14
M=250	4467.68	5305.90	4933.73	4802.01	6320.48	4300.47	15.80	9.45	6.96	29.31	-3.89
M=300	5941.34	7237.45	6595.24	6430.20	8740.22	5692.03	17.91	9.91	7.60	32.02	-4.38
M=350	8001.90	10011.13	8952.02	8800.23	11785.94	7608.50	20.07	10.61	9.07	32.11	-5.17
M=400	10923.48	13984.74	12285.24	12133.89	16193.65	10322.54	21.89	11.08	9.98	32.54	-5.82
M=450	14626.22	18936.15	16778.60	16532.69	23525.89	13755.50	22.76	12.83	11.53	37.83	-6.33
M=500	19154.32	24920.18	22111.14	21888.92	32393.80	17798.11	23.14	13.37	12.49	40.87	-7.62
M=550	25045.93	34428.09	30404.68	29740.56	43914.02	23130.77	27.25	17.62	15.79	42.97	-8.28
M=600	28169.45	40907.70	35731.93	34112.71	52706.36	25786.77	31.14	21.16	17.42	46.55	-9.24
M=650	31169.31	48516.32	42353.93	41425.40	66321.02	28125.51	35.75	26.41	24.76	53.00	-10.82
M=700	35410.88	57826.99	50744.96	49042.35	95915.25	31772.89	38.76	30.22	27.80	63.08	-11.45
M=750	39592.90	71565.05	61404.83	58852.53	131697.96	35365.70	44.68	35.52	32.73	69.94	-11.95
M=800	44028.95	84699.67	73191.12	68646.90	181529.99	39123.26	48.02	39.84	35.86	75.75	-12.54
M=850	48456.20	103430.29	84408.33	77638.61	305945.03	42853.56	53.15	42.59	37.59	84.16	-13.07
M=900	53824.78	123860.29	96256.60	90949.70	349609.11	47077.33	56.54	44.08	40.82	84.60	-14.33
M=950	59986.88	150599.42	120528.78	108239.48	454767.61	52312.03	60.17	50.23	44.58	86.81	-14.67
M=1000	66841.38	202659.29	157651.30	135268.22	1006295.31	58029.83	67.02	57.60	50.59	93.36	-15.18
Avg.	26493.94	53040.07	43823.94	40667.53	147259.41	23715.40	33.40	23.84	21.10	51.48	-8.56

TABLE VII
RUNTIME OVERHEADS OF TASK OFFLOADING ALGORITHMS WHEN RUNNING REAL-LIFE APPLICATIONS

The Number of end devices	Measured runtime (seconds) of six algorithms						Speedup (times) of our approach against peers				
	Proposed	ELYO	ELGT	ELDM	ERPS	EILP	ELYO	ELGT	ELDM	ERPS	EILP
M=100	4.23	35.10	59.65	56.13	92.28	155.76	7.30	13.11	12.28	20.83	35.84
M=150	4.75	44.82	67.50	56.06	94.13	210.93	8.44	13.21	10.81	18.82	43.42
M=200	5.13	61.10	84.12	85.65	106.44	264.96	10.90	15.39	15.68	19.73	50.61
M=250	6.88	112.93	134.01	100.25	178.84	399.38	15.41	18.47	13.57	24.99	57.04
M=300	7.41	136.88	115.23	118.63	175.22	599.86	17.48	14.55	15.01	22.65	79.97
M=350	8.77	164.58	163.42	172.93	253.55	799.35	17.77	17.63	18.72	27.91	90.14
M=400	9.51	216.95	194.08	189.29	278.72	929.33	21.82	19.42	18.91	28.32	96.76
M=450	10.78	233.80	234.10	257.56	343.03	1206.57	20.69	20.72	22.89	30.82	110.93
M=500	11.84	307.17	318.70	254.51	367.35	1242.78	24.94	25.92	20.50	30.03	103.97
M=550	12.96	383.16	386.44	363.57	384.44	1460.10	28.57	28.83	27.06	28.67	111.70
M=600	13.31	426.79	431.61	430.66	508.42	1749.56	31.07	31.44	31.36	37.21	130.48
M=650	13.99	485.05	533.86	507.33	594.13	1787.63	33.68	37.17	35.27	41.48	126.81
M=700	14.53	430.68	569.44	605.24	659.70	2083.21	28.64	38.19	40.66	44.41	142.38
M=750	16.34	572.14	866.25	699.82	703.90	2740.59	34.02	52.02	41.83	42.08	166.74
M=800	17.80	725.37	873.25	736.05	856.84	2924.97	39.75	48.06	40.35	47.14	163.32
M=850	18.51	734.42	941.62	857.25	951.30	3322.22	38.67	49.86	45.31	50.39	178.46
M=900	19.63	850.45	1290.77	1074.40	887.49	4198.74	42.33	64.76	53.74	44.22	212.92
M=950	20.12	851.04	1234.65	1181.88	1160.41	4369.73	41.29	60.35	57.73	56.67	216.15
M=1000	22.75	1266.82	1434.50	1641.83	1570.71	5215.48	54.69	62.06	71.18	68.05	228.29
Avg.	12.59	423.12	522.80	494.16	535.10	1876.90	27.24	33.22	31.20	36.02	123.47

TABLE VIII
SCHEDULABILITY WHEN ONLY CONSIDERING TASK DEPENDENCE/DEADLINE CONSTRAINTS OF REAL-LIFE APPLICATIONS

The Number of end devices	Schedulability when only considering inter- task dependence constraints						Schedulability when only considering task deadline constraints					
	Proposed	ELYO	ELGT	ELDM	ERPS	EILP	Proposed	ELYO	ELGT	ELDM	ERPS	EILP
M=100	100%	57%	75%	55%	100%	100%	100%	51%	70%	57%	86%	100%
M=150	100%	51%	72%	49%	100%	100%	100%	49%	65%	51%	81%	100%
M=200	100%	50%	70%	48%	100%	100%	100%	45%	63%	49%	80%	100%
M=250	100%	47%	69%	47%	100%	100%	100%	44%	60%	45%	79%	100%
M=300	100%	46%	68%	47%	100%	100%	100%	44%	58%	43%	77%	100%
M=350	100%	44%	67%	46%	100%	100%	100%	40%	57%	42%	76%	100%
M=400	100%	43%	66%	44%	100%	100%	100%	39%	56%	42%	75%	100%
M=450	100%	43%	63%	42%	100%	100%	100%	37%	55%	41%	70%	100%
M=500	100%	41%	60%	37%	100%	100%	100%	37%	52%	40%	65%	100%
M=550	100%	39%	57%	37%	100%	100%	100%	33%	50%	37%	63%	100%
M=600	100%	36%	54%	36%	100%	100%	100%	30%	46%	35%	60%	100%
M=650	100%	36%	53%	35%	100%	100%	100%	29%	45%	31%	57%	100%
M=700	100%	35%	50%	31%	100%	100%	100%	28%	43%	31%	55%	100%
M=750	100%	32%	47%	27%	100%	100%	100%	28%	42%	30%	54%	100%
M=800	100%	29%	46%	22%	100%	100%	100%	25%	40%	30%	50%	100%
M=850	100%	28%	42%	22%	100%	100%	100%	25%	39%	26%	46%	100%
M=900	100%	28%	39%	21%	100%	100%	100%	23%	33%	24%	42%	100%
M=950	100%	21%	31%	21%	100%	100%	100%	20%	30%	22%	39%	100%
M=1000	100%	17%	24%	18%	100%	100%	100%	17%	27%	19%	33%	100%
Avg.	100%	38%	55%	36%	100%	100%	100%	34%	49%	37%	62%	100%

B. Investigation Results

We first investigate the energy consumption of six task offloading methods when running real-life DAG applications. We observe from Table VI that our approach achieves 33.40%, 23.84%, 21.10%, and 51.48% energy savings on average compared with peer algorithms ELYO, ELGT, ELDM, and ERPS, respectively. More importantly, our approach exhibits an accelerating trend in terms of energy savings as the number of end devices in the system increases. The main reason is that our approach fully exploits the heterogeneity of DAG applications in energy demands, whereas the differentiated energy requirements

TABLE IX
SCHEDULABILITY WHEN CONSIDERING TASK RELIABILITY AND ALL CONSTRAINTS OF REAL-LIFE APPLICATIONS

The Number of end devices	Schedulability when only considering task reliability constraints						Schedulability when considering all design constraints					
	Proposed	ELYO	ELGT	ELDM	ERPS	EILP	Proposed	ELYO	ELGT	ELDM	ERPS	EILP
M=100	100%	62%	53%	61%	90%	100%	100%	50%	50%	47%	83%	100%
M=150	100%	60%	53%	60%	82%	100%	100%	44%	48%	42%	80%	100%
M=200	100%	57%	52%	59%	77%	100%	100%	41%	47%	41%	73%	100%
M=250	100%	56%	52%	57%	73%	100%	100%	40%	47%	40%	72%	100%
M=300	100%	54%	51%	55%	70%	100%	100%	38%	45%	38%	68%	100%
M=350	100%	53%	51%	54%	67%	100%	100%	37%	45%	37%	66%	100%
M=400	100%	52%	50%	54%	63%	100%	100%	37%	44%	34%	60%	100%
M=450	100%	50%	44%	51%	60%	100%	100%	36%	42%	34%	58%	100%
M=500	100%	48%	41%	51%	58%	100%	100%	34%	40%	32%	54%	100%
M=550	100%	44%	40%	47%	57%	100%	100%	30%	39%	31%	53%	100%
M=600	100%	43%	36%	44%	54%	100%	100%	29%	35%	29%	52%	100%
M=650	100%	40%	35%	43%	52%	100%	100%	26%	34%	28%	51%	100%
M=700	100%	37%	33%	42%	51%	100%	100%	25%	29%	28%	47%	100%
M=750	100%	33%	30%	39%	49%	100%	100%	24%	28%	23%	43%	100%
M=800	100%	30%	29%	38%	45%	100%	100%	22%	28%	22%	40%	100%
M=850	100%	25%	28%	34%	43%	100%	100%	20%	21%	18%	34%	100%
M=900	100%	24%	27%	32%	40%	100%	100%	20%	18%	17%	33%	100%
M=950	100%	18%	19%	26%	36%	100%	100%	18%	15%	12%	30%	100%
M=1000	100%	14%	14%	23%	30%	100%	100%	13%	12%	11%	25%	100%
Avg.	100%	42%	39%	46%	58%	100%	100%	31%	35%	30%	54%	100%

of DAG applications are largely ignored by all peer algorithms. In addition, we witness that the energy consumption of our approach is 8.56% higher on average than that of benchmarking algorithm EILP. This is because benchmarking algorithm EILP derives globally optimal computation offloading solutions by exploiting the ILP technique.

We then investigate the runtime of six task offloading algorithms when running real-life DAG applications. As demonstrated in Table VII, our approach achieves 27.24, 33.22, 31.20, 36.02, and 123.47 times of runtime speedup on average compared with peer algorithms ELYO, ELGT, ELDM, ERPS, and EILP, respectively. As aforementioned, the reason is that our approach can search for energy-aware reliability-ensured task offloading solutions for differentiated DAG groups in parallel with the help of ISODATA clustering technique.

We finally investigate the schedulability of six task offloading algorithms when running real-life DAG applications and summarize the comparison results in Tables VIII and IX. Clearly, the results are consistent with the observations for synthetic DAG applications. That is, except for our approach and benchmarking algorithm EILP, all other algorithms (i.e., ELYO, ELGT, ELDM, and ERPS) cannot guarantee 100% schedulability for real-life applications. Meanwhile, our approach yields a better tradeoff between the holistic energy savings of end devices and the runtime overheads compared with benchmarking algorithm EILP.

IX. CONCLUSION

This article aimed to address the problem of energy minimization under DAG timing, precedence, and reliability constraints in end–end–edge collaborated large-scale CPS. To this end, our approach first utilized a clustering method to distinguish differentiated energy demands of DAG applications. Then, our approach developed a PID-based task offloading scheme for computation-intensive DAG clusters and a PSOSR-based task offloading scheme for communication-intensive DAG clusters. In the future work, we plan to extend the current study from the following three aspects.

1) Integrate the popular dynamic voltage and frequency scaling technique into end devices for energy optimization.
2) Consider the modern end devices powered by renewable generations, e.g., solar energy.
3) Investigate the approximate computing requirements of DAG applications.

APPENDIX

For the sake of easy presentation, let θ denote the sorted processor-dependent parameters in ascending order, i.e., $\theta = [\theta_1, \theta_2, \ldots, \theta_{|\Phi_{w_k}|}]$ satisfying $\theta_1 \leq \theta_2 \cdots \leq \theta_{|\Phi_{w_k}|}$. Similarly, let $\mathcal{Y}$ denote the sorted task-dependent parameters in descending order, i.e., $\mathcal{Y} = [\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_{|\Phi_{w_k}|}]$ satisfying $\mathcal{Y}_1 \geq \mathcal{Y}_2 \geq \cdots \geq \mathcal{Y}_{|\Phi_{w_k}|}$. At this moment, the computation energy consumption $E(\Omega'_{w_k})$ is given by

$$E(\Omega'_{w_k}) = \theta_1 \times \mathcal{Y}_1 + \cdots + \theta_{|\Phi_{w_k}|} \times \mathcal{Y}_{|\Phi_{w_k}|}. \quad (33)$$

Suppose that the locations of two elements $\mathcal{Y}_i$ and $\mathcal{Y}_j$ $(i < j)$ in the matrix $\mathcal{Y}$ are swapped with each other, i.e., $\mathcal{Y}_{\text{swap}} = [\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_{i-1}, \mathcal{Y}_j, \mathcal{Y}_{i+1}, \ldots, \mathcal{Y}_{j-1}, \mathcal{Y}_i, \mathcal{Y}_{j+1}, \ldots, \mathcal{Y}_{|\Phi_{w_k}|}]$. Then, the computation energy $E_{\text{swap}}(\Omega'_{w_k})$ is given by

$$E_{\text{swap}}(\Omega'_{w_k}) = \theta_1 \times \mathcal{Y}_1 + \theta_2 \times \mathcal{Y}_2 + \cdots + \theta_i \times \mathcal{Y}_j + \cdots + \theta_j \times \mathcal{Y}_i + \cdots + \theta_{|\Phi_{w_k}|} \times \mathcal{Y}_{|\Phi_{w_k}|}. \quad (34)$$

By comparing the computation energy consumption $E(\Omega'_{w_k})$ and $E_{\text{swap}}(\Omega'_{w_k})$, we obtain (35) due to the optimality assumption of energy consumption $E(\Omega'_{w_k})$, that is,

$$E_{\text{swap}}(\Omega'_{w_k}) - E(\Omega'_{w_k}) = (\theta_i - \theta_j)(\mathcal{Y}_j - \mathcal{Y}_i) \geq 0. \quad (35)$$

Considering the inequality $i < j$ holds, we deduce that $\mathcal{Y}_i$ should be no less than $\mathcal{Y}_j$, i.e., the inequality $\mathcal{Y}_i \geq \mathcal{Y}_j$ holds. Based on the aforementioned analysis, we can iteratively exchange the position of any two elements in matrix $\mathcal{Y}$. In every iteration of the element position swapping, it is clear that the inequality

$\mathcal{Y}_i \geq \mathcal{Y}_j$ holds for $i < j$. At the end of the whole swapping procedure, the inequality $\mathcal{Y}_1 \geq \mathcal{Y}_2 \geq \cdots \geq \mathcal{Y}_{|\Phi_{w_k}|}$ holds.

REFERENCES

[1] K. Cao, S. Hu, Y. Shi, A. Colombo, S. Karnouskos, and X. Li, "A survey on edge and edge-cloud computing assisted cyber-physical systems," *IEEE Trans. Ind. Inform.*, vol. 17, no. 11, pp. 7806–7819, Nov. 2021.

[2] P. Zhou, D. Zuo, K. Hou, Z. Zhang, and J. Dong, "Improving the dependability of self-adaptive cyber physical system with formal compositional contract," *IEEE Trans. Rel.*, vol. 69, no. 3, pp. 1130–1146, Sep. 2020.

[3] K. Cao, T. Wei, M. Chen, K. Li, J. Weng, and W. Tan, "Exploring reliable edge-cloud computing for service latency optimization in sustainable cyber-physical systems," *Software, Pract. Experience*, vol. 51, no. 11, pp. 2225–2237, 2021.

[4] M. Sun, X. Xu, Y. Huang, Q. Wu, X. Tao, and P. Zhang, "Resource management for computation offloading in D2D-aided wireless powered mobile-edge computing networks," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8005–8020, May 2021.

[5] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[6] K. Wang, Y. Zhou, J. Li, L. Shi, W. Chen, and L. Hanzo, "Energy-efficient task offloading in massive MIMO-aided multi-pair fog-computing networks," *IEEE Trans. Commun.*, vol. 69, no. 4, pp. 2123–2137, Apr. 2021.

[7] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4188–4200, Jun. 2019.

[8] Y. Kai, J. Wang, and H. Zhu, "Energy minimization for D2D-assisted mobile edge computing networks," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[9] Y. Yang, C. Long, J. Wu, S. Peng, and B. Li, "D2D-enabled mobile-edge computation offloading for multi-user IoT network," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12490–12504, Aug. 2021.

[10] M. Sun, X. Xu, X. Tao, and P. Zhang, "Large-scale user-assisted multi-task online offloading for latency reduction in D2D-enabled heterogeneous networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2456–2467, Oct.–Dec. 2020.

[11] A. Naithani, S. Eyerman, and L. Eeckhout, "Optimizing soft error reliability through scheduling on heterogeneous multicore processors," *IEEE Trans. Comput.*, vol. 67, no. 6, pp. 830–846, Jun. 2018.

[12] A. Ansari, J. Saber-Latibari, M. Pasandideh, and A. Ejlali, "Simultaneous management of peak-power and reliability in heterogeneous multicore embedded systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 623–633, Mar. 2020.

[13] L. Li et al., "Game theoretic feedback control for reliability enhancement of EtherCAT-based networked systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1599–1610, Sep. 2019.

[14] A. Savino, A. Vallero, and S. Carlo, "ReDO: Cross-layer multi-objective design-exploration framework for efficient soft error resilient systems," *IEEE Trans. Comput.*, vol. 67, no. 10, pp. 1462–1477, Oct. 2018.

[15] B. Hu, Z. Cao, and M. Zhou, "Energy-minimized scheduling of real-time parallel workflows on heterogeneous distributed computing systems," *IEEE Trans. Serv. Comput.*, vol. 15, no. 5, pp. 2766–2779, Sep./Oct. 2022.

[16] K. Cao, Y. Cui, Z. Liu, W. Tan, and J. Weng, "Edge intelligent joint optimization for lifetime and latency in large-scale cyber-physical system," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 22267–22279, Nov. 2022.

[17] G. Premsankar, M. Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[18] H. Huang, V. Chaturvedi, G. Quan, J. Fan, and M. Qiu, "Throughput maximization for periodic real-time systems under the maximal temperature constraint," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 2s, pp. 1–22, 2014.

[19] Q. Zhu, H. Tang, J. Huang, and Y. Hou, "Task scheduling for multi-cloud computing subject to security and reliability constraints," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 4, pp. 848–865, Apr. 2021.

[20] K. Cao, Y. Cui, L. Li, J. Zhou, and S. Hu, "CPU-GPU cooperative QoS optimization of personalized digital healthcare using machine learning and swarm intelligence," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 2022, to be published, doi: 10.1109/TCBB.2022.3207509.

[21] X. Dai et al., "Task co-offloading for D2D-assisted mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 480–490, Jan. 2023.

[22] X. Zhu and M. Zhou, "Multiobjective optimized cloudlet deployment and task offloading for mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 20, pp. 15582–15595, Oct. 2021.

[23] H. Santos and T. Toffolo, "Python MIP Tools," Accessed: May 17, 2022. [Online]. Available: https://pypi.org/project/mip/

[24] J. Liang, K. Li, C. Liu, and K. Li, "Joint offloading and scheduling decisions for DAG applications in mobile edge computing," *Neurocomputing*, vol. 424, pp. 160–171, 2021.

[25] M. Sabin, "Convergence and consistency of fuzzy c-means/ISODATA algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-5, no. 5, pp. 661–668, Sep. 1987.

[26] J. Bi, H. Yuan, J. Zhai, M. Zhou, and H. Poor, "Self-adaptive Bat algorithm with genetic operations," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 7, pp. 1284–1294, Jul. 2022.

[27] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[28] F. Wang, H. Zhang, and A. Zhou, "A particle swarm optimization algorithm for mixed-variable optimization problems," *Swarm Evol. Comput.*, vol. 60, pp. 1–12, 2021.

[29] T. Wei, X. Chen, and S. Hu, "Reliability-driven energy efficient task scheduling for multiprocessor real-time systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 10, pp. 1569–1573, Oct. 2011.

[30] Dell Inc., "Dell PowerEdge R930 Server," Accessed: Mar. 10, 2023. [Online]. Available: https://i.dell.com/sites/csdocuments/Shared-Content_data-Sheets_Documents/en/aa/PowerEdge_R930_spec_sheet-FINAL.pdf

[31] M. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 813–825, Mar. 2017.

[32] K. Cao et al., "Affinity-driven modeling and scheduling for makespan optimization in heterogeneous multiprocessor systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1189–1202, Jul. 2019.

[33] R. Dick, D. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proc. IEEE Int. Workshop Hardware/Software Codesign*, 1998, pp. 97–101.

[34] A. Masood, S. Ahmad, H. Khan, and E. Munir, "Network reconfiguration algorithm (NRA) for scheduling communication-intensive graphs in heterogeneous computing environment," *Cluster Comput.*, vol. 23, no. 2, pp. 1419–1438, 2020.

[35] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2867–2876, Nov. 2014.

[36] L. Bittencourt, R. Sakellariou, and E. Madeira, "DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm," in *Proc. Euromicro Conf. Parallel, Distrib. Netw.-Based Process.*, 2010, pp. 27–34.

[37] J. Zhang, X. Zhou, T. Ge, X. Wang, and H. Hwang, "Joint task scheduling and containerizing for efficient edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 2086–2100, Aug. 2021.

[38] H. Li, J. Wu, Z. Jiang, X. Li, and X. Wei, "Minimum backups for stream processing with recovery latency guarantees," *IEEE Trans. Rel.*, vol. 66, no. 3, pp. 783–794, Sep. 2017.

Kun Cao (Member, IEEE) received the Ph.D. degree in computer science from East China Normal University, Shanghai, China, in 2020.

He is currently an Associate Professor with the College of Information Science and Technology, Jinan University, Guangzhou, China. His current research interests include the areas of Internet of things, edge/fog/cloud computing, and cyber-physical systems.

Dr. Cao was the recipient of the Young Professional Award from IEEE Technical Committee on Secure and Dependable Measurement (TCSDM) in 2022. He has been serving as an Associate Editor for *Journal of Circuits, Systems, and Computers*.

Jian Weng (Member, IEEE) received the Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2008 .

He is currently a Full Professor with the College of Information Science and Technology, Jinan University, Guangzhou, China. His current research interests include information security and artificial intelligence.

Dr. Weng has served as an Associate Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

Keqin Li (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Houston, Houston, Texas, USA, in 1990. He is a SUNY Distinguished Professor of computer science with the State University of New York, New Paltz, NY, USA. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, Internet of things, and cyber-physical systems.

Dr. Li is currently serving or has served on the editorial boards of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, and IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.