

Optimal Multi-Constrained Workflow Scheduling for Cyber-Physical Systems in the Edge-Cloud Continuum

Andreas Kouloumpris, Georgios L. Stavrinides, Maria K. Michael, and Theocharis Theocharides

KIOS Research and Innovation Center of Excellence

Department of Electrical and Computer Engineering, University of Cyprus

Nicosia, Cyprus

Email: {kouloumpris.andreas, stavrinides.georgios, mmichael, ttheocharides}@ucy.ac.cy

Abstract—The emerging edge-hub-cloud paradigm has enabled the development of innovative latency-critical cyber-physical applications in the edge-cloud continuum. However, this paradigm poses multiple challenges due to the heterogeneity of the devices at the edge of the network, their limited computational, communication, and energy capacities, as well as their different sensing and actuating capabilities. To address these issues, we propose an optimal scheduling approach to minimize the overall latency of a workflow application in an edge-hub-cloud cyber-physical system. We consider multiple edge devices cooperating with a hub device and a cloud server. All devices feature heterogeneous multicore processors and various sensing, actuating, or other specialized capabilities. We present a comprehensive formulation based on continuous-time mixed integer linear programming, encapsulating multiple constraints often overlooked by existing approaches. We conduct a comparative experimental evaluation between our method and a well-established and effective scheduling heuristic, which we enhanced to consider the constraints of the specific problem. The results reveal that our technique outperforms the heuristic, achieving an average latency improvement of 13.54% in a relevant real-world use case, under varied system configurations. In addition, the results demonstrate the scalability of our method under synthetic workflows of varying sizes, attaining a 33.03% average latency decrease compared to the heuristic.

Index Terms—scheduling, workflow, cyber-physical system, edge-cloud, mixed integer linear programming, optimization.

I. INTRODUCTION

The edge-hub-cloud paradigm has emerged as an alternative to the conventional edge computing concept. This shift is driven by the increasing demand for intelligent decision-making and real-time response at the network edge [1]–[4]. In contrast to the traditional concept that typically has three distinct layers within the edge-cloud continuum (i.e., edge devices, edge servers, and cloud servers), this paradigm considers edge and hub devices that are usually battery-powered in the bottom layer, and cloud servers in the top layer. A hub device (e.g., a smartphone or laptop) often has a higher computational capacity than an edge device (e.g., a wearable or single-board computer). Furthermore, it is physically closer to the edge devices than an edge server (although less capable), facilitating their communication with the remote cloud data center.

With the advent of the edge-hub-cloud paradigm, novel cyber-physical applications have been developed, consisting of precedence-constrained tasks that require diverse device

capabilities for their execution, such as specific sensors, actuators, or software/hardware modules [5]. These applications, referred to as workflows, encompass a wide range of latency-critical scenarios. The use of biomedical edge devices with integrated sensors and actuators for remote patient monitoring and support (e.g., smart pacemakers and wearable cardioverter defibrillators) [1], [2] or the use of unmanned aerial vehicles (UAVs) with different capabilities for autonomous critical infrastructure inspection or search-and-rescue missions (e.g., UAVs with thermal/multispectral cameras or customized payload release systems) [3], [4], are some prominent examples. In these scenarios, multiple edge devices with distinct capabilities cooperate with a hub device and a cloud server, which may also feature specialized software or hardware components, like libraries or accelerators for machine learning inference.

The intrinsic criticality of such applications requires *optimal* task scheduling to determine where to allocate and when to execute each task, so as to achieve the minimum possible latency, which is usually bounded by a strict deadline [6]. In contrast to heuristic approaches, exact methods such as mixed integer linear programming (MILP) can yield optimal schedules, but are typically computationally intensive. However, the pre-programmed nature of the examined applications allows the use of these methods for offline scheduling—although solutions should still be provided in a reasonable time to be practical. In exact methods, time is modeled as either discrete or continuous. Discrete-time models simplify the problem by considering that events occur only at the beginning or end of predefined time intervals, which is not a realistic assumption. On the other hand, continuous-time models offer higher accuracy by allowing events to occur at any time. However, this flexibility makes the problem formulation more challenging, particularly the modeling of time-dependent cumulative constraints for the concurrent utilization of limited resources by multiple tasks, such as main memory, storage, or specific device capabilities [7].

Moreover, challenges arise due to the heterogeneity of the devices where the considered applications are deployed, as they often feature different multicore processors and varied sensing/actuating capabilities. These challenges are amplified

by the diverse communication characteristics of the operating setting. In addition, moving from the cloud towards the network edge, there are increasing limitations in the computational, memory, storage, and energy capacities of the devices. In such environments, it is imperative to employ a workflow scheduling strategy that takes into account all these constraints, which make finding an optimal schedule particularly challenging. To this end, we propose an offline continuous-time MILP approach to optimally schedule a workflow application in an edge-hub-cloud cyber-physical system (CPS) with multiple edge devices, heterogeneous multicore processors, and distinct device capabilities. Our objective is to minimize the overall latency, while considering crucial constraints that characterize these systems.

The contributions of this work are summarized as follows:

- We propose a comprehensive continuous-time MILP formulation to optimally schedule a workflow application in the examined edge-hub-cloud CPS by minimizing the overall latency. Our formulation is facilitated by an extended representation of the application task graph.
- We holistically address multiple constraints that are overlooked by existing scheduling methods, both exact and heuristic. These constraints are based on the memory, storage, and energy limitations of the devices, the heterogeneity and multicore architecture of the processors, the diverse device capabilities, the execution deadline, as well as the computational and communication latency and energy requirements of the tasks.
- As the proposed approach is the first to provide an optimal solution for the considered problem, constraints, and architecture, we evaluate and compare it against a widely used and effective scheduling heuristic, the heterogeneous earliest finish time (HEFT) [8]–[10]. To ensure a fair and meaningful comparison, we extended HEFT to incorporate the constraints of the examined problem.
- In our experiments, we consider a real-world workflow for the autonomous UAV-based power infrastructure inspection, under various system configurations. To further validate and investigate the scalability of our method to applications of different sizes, we utilize multiple synthetic workflows with appropriate parameters.

The rest of the paper is organized as follows. Section II provides an overview of related literature. Section III describes the proposed MILP approach, whereas Section IV explains our extension to HEFT. Section V presents the experimental setup, the evaluation results, and the empirical scalability analysis of our technique. Section VI provides concluding remarks.

II. RELATED WORK

Workflow scheduling in distributed environments is a well-studied problem that has been investigated extensively using both exact and heuristic methods [8]–[18]. For example, MILP-based scheduling techniques for deadline-constrained workflows are proposed in [11]–[15], with [13] and [15] focusing on specific CPS architectures. While these methods [11]–[15] provide optimal schedules, none of them considers a

TABLE I
COMPARISON WITH RELATED RESEARCH EFFORTS

Reference	Considered Constraints					Optimal Solution (d/c) ²	Multi-tier Environ.	Multicore Processors (h/H) ³
	Deadline	Capability (s/m) ¹	Memory	Storage	Energy			
[8]	-	-	-	-	-	-	-	✓(H)
[9]	-	-	-	-	-	-	-	✓(H)
[10]	-	-	-	-	-	-	-	✓(H)
[11]	✓	-	-	-	-	✓(d)	-	✓(H)
[12]	✓	-	-	-	✓	✓(c)	-	✓(h)
[13]	✓	✓(s)	-	-	✓	✓(c)	-	-
[14]	✓	-	-	-	✓	✓(c)	-	✓(H)
[15]	✓	-	-	-	✓	✓(d)	-	-
[16]	✓	✓(m)	-	-	-	-	-	✓(H)
[17]	✓	-	-	-	-	-	✓	✓(H)
[18]	✓	-	-	-	-	-	✓	✓(H)
This work	✓	✓(m)	✓	✓	✓	✓(c)	✓	✓(H)

¹single/multiple capabilities (e.g., sensors and/or actuators) per CPS device.

²discrete-time/continuous-time approach. ³homogeneous/Heterogeneous processors.

multi-tier setting or the memory and storage limitations of the devices. The approaches in [13] and [15] also assume single-core processors, whereas homogeneous (rather than heterogeneous) multicore processors are examined in [12]. Moreover, only [12]–[14] address energy consumption. Even though the technique in [13] considers different sensing/actuating capabilities among CPS devices, it is limited to only one sensor or actuator per device. On the other hand, no device capabilities are considered in [15], although a CPS is examined. Regarding the representation of time in the problem formulation, the methods in [11] and [15] employ discrete-time MILP, in contrast to [12]–[14] that use continuous-time MILP.

In addition to exact approaches, heuristic techniques have also received significant attention for workflow scheduling in CPS and multi-tier architectures [16]–[18]. For instance, the heuristic in [16] aims to schedule tasks with different sensing/actuating requirements in a CPS with multiple sensors and actuators per device. Moreover, a two-stage strategy is proposed in [17], combining a heuristic search approach and task clustering to schedule workflow applications in an edge-cloud environment. In the same context, a two-phase heuristic is presented in [18], which makes scheduling decisions based on the computation and communication requirements of each task in a three-tier system. Even though these methods are suitable for heterogeneous multicore processors and workflows with execution deadlines, they do not consider the specific environment, nor all the constraints examined in this work.

Among the most well-established and effective workflow scheduling heuristics is the heterogeneous earliest finish time (HEFT) [8]. It remains one of the most frequently used techniques for distributed heterogeneous (multicore) processors, providing high-quality schedules [9]. For example, it is employed in [10] to schedule embedded vision applications in a multicore architecture. Similarly, it is used in [9] to schedule dataflow applications in a safety-critical embedded system. However, HEFT does not support the deadline, capability, memory, storage, and energy constraints arising from the system setting considered in this work. Moreover, as it is a heuristic method, it cannot guarantee optimal solutions.

Overall, existing exact and heuristic workflow scheduling methods do not comprehensively address all the constraints considered in the proposed MILP approach, nor examine

the specific CPS architecture. The qualitative comparison of this work with the presented research efforts is summarized in Table I. Although we previously investigated edge-hub-cloud systems in [19]–[22], we did not consider multiple edge devices, nor a CPS with different sensing, actuating, or other specialized capabilities. More importantly, our previous approaches pose a key limitation, as they can only determine where to allocate tasks, but not when to execute them, performing only the mapping step of the scheduling process.

III. PROPOSED MILP APPROACH

A. Application Model

The workflow application comprises a set of α tasks $\mathcal{T} = \{\tau_i \mid 1 \leq i \leq \alpha\}$. It is represented by a task graph (TG) in the form of a directed acyclic graph $G = (\mathcal{N}, \mathcal{A})$ [10]. $\mathcal{N} = \{N_i \mid \tau_i \in \mathcal{T}\}$ is the set of its nodes, whereas $\mathcal{A} = \{A_{i \rightarrow j} \mid N_i, N_j \in \mathcal{N}, i \neq j, \exists \text{ a data dependency } N_i \rightarrow N_j\}$ is the set of its arcs. A node $N_i \in \mathcal{N}$ represents a task $\tau_i \in \mathcal{T}$. A task is an indivisible unit of work. Tasks are considered to be non-preemptive, as preemption in time-constrained applications may lead to performance degradation [23]. An arc $A_{i \rightarrow j} \in \mathcal{A}$ between two nodes N_i and N_j (corresponding to parent task τ_i and child task τ_j , respectively), denotes the communication and precedence relationship between the two tasks. There is a predefined deadline (i.e., latency threshold) L_{thr} before which all tasks of the application should be completed. L_{thr} also denotes the time horizon of the examined scheduling problem.

B. System Model

We consider a CPS comprising a set of β different devices $\mathcal{U} = \{u_{\lambda k} \mid \lambda \in \{e, h, c\}, 1 \leq k \leq \beta_\lambda, \sum_{\lambda \in \{e, h, c\}} \beta_\lambda = \beta\}$. λ denotes the type of the device (e represents an edge device, h a hub device, and c a cloud server), whereas k and β_λ are the device index and the number of devices of the particular type, respectively. All devices in $\mathcal{U}$ are multicore. For the execution of the specific application, we reserve on each device $u_{\lambda k} \in \mathcal{U}$ a set of $\gamma_{\lambda k}$ processing cores $\mathcal{P}_{\lambda k} = \{p_{\lambda k, q} \mid 1 \leq q \leq \gamma_{\lambda k}\}$ (q denotes the processing core index). The set of all reserved processing cores on all system devices is defined as $\mathcal{P} = \bigcup_{u_{\lambda k} \in \mathcal{U}} \mathcal{P}_{\lambda k}$. The processing cores in $\mathcal{P}$ may be heterogeneous. Thus, the latency and power consumption required to execute a given task may vary. Each processing core can execute one task at a time. As the system resources are shared with other applications, for the execution of the examined workflow we consider memory, storage, and energy budgets for each device $u_{\lambda k}$. These budgets, denoted by $M_{\lambda k}^{\text{bgt}}$, $S_{\lambda k}^{\text{bgt}}$, and $E_{\lambda k}^{\text{bgt}}$, respectively, are shared among the reserved processing cores of each device.

Furthermore, we consider a set of δ device capabilities $\mathcal{C} = \{c_a \mid 0 \leq a < \delta\}$. c_0 represents the basic computational capability of a device, whereas c_a with $a > 0$ denotes a specialized capability (in addition to the device's basic computational capability), such as a specific sensor, actuator, or software/hardware module. Each device $u_{\lambda k} \in \mathcal{U}$ has a set of capabilities $\mathcal{C}_{\lambda k} \subseteq \mathcal{C}$, such that $c_0 \in \mathcal{C}_{\lambda k}$. The binary parameter $y_{\lambda k}^{c_a}$ indicates whether device $u_{\lambda k}$ features capability $c_a \in \mathcal{C}$ ($y_{\lambda k}^{c_a} = 1$) or not ($y_{\lambda k}^{c_a} = 0$). Accordingly, the binary

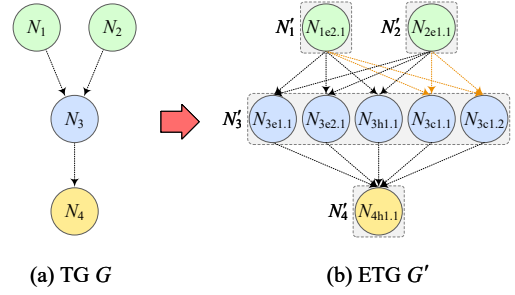


Fig. 1. Transformation example: (a) the initial TG and (b) the final ETG.

parameter $z_i^{c_a}$ denotes if task τ_i requires capability $c_a \in \mathcal{C}$ ($z_i^{c_a} = 1$) or not ($z_i^{c_a} = 0$). We consider that each task $\tau_i \in \mathcal{T}$ requires one device capability [16].

Each pair of devices can communicate directly or indirectly through another device. Specifically, the communication channel between devices $u_{\lambda k}$ and $u_{\mu l}$ is defined as $\xi_{\lambda k, \mu l} = \langle \theta_{\lambda k, \mu l}, \pi_{\lambda k, \mu l}, \rho_{\lambda k, \mu l}, \mathcal{I}_{\lambda k, \mu l} \rangle$, where $\theta_{\lambda k, \mu l}$ denotes its bandwidth, whereas $\pi_{\lambda k, \mu l}$ and $\rho_{\lambda k, \mu l}$ denote the energy consumption for the transmission and reception of a unit of data over the particular channel, respectively. If $u_{\lambda k}$ and $u_{\mu l}$ can only communicate through device $u_{\nu m}$, then $\mathcal{I}_{\lambda k, \mu l} = \{u_{\nu m}\}$, otherwise $\mathcal{I}_{\lambda k, \mu l} = \emptyset$.

C. Extended Task Graph

To facilitate the MILP formulation of the considered problem, we transform the initial TG $G = (\mathcal{N}, \mathcal{A})$ of the application into an extended task graph (ETG) $G' = (\mathcal{N}', \mathcal{A}')$, based on the main principles of our approach in [19]. The ETG encapsulates both the application and system models, incorporating their communication and energy aspects. In particular, each node $N_i \in \mathcal{N}$ in G is transformed into a composite node $N'_i \in \mathcal{N}'$ in G' , such that:

$$N'_i = \{N_{i\lambda k, q} \mid p_{\lambda k, q} \in \mathcal{P}, y_{\lambda k}^{c_a} z_i^{c_a} = 1, c_a \in \mathcal{C}\}. \quad (1)$$

Hence, N'_i is the set of nodes corresponding to the possible allocations of task τ_i on the reserved processing cores of the devices featuring its required capability. An individual node $N_{i\lambda k, q} \in N'_i$ (denoting a specific allocation of τ_i) is referred to as a *candidate node*.

Accordingly, each arc $A_{i \rightarrow j} \in \mathcal{A}$ in G is transformed into a composite arc $A'_{i \rightarrow j} \in \mathcal{A}'$ in G' , such that:

$$A'_{i \rightarrow j} = \{A_{i\lambda k, q \rightarrow j\mu l, r} = N_{i\lambda k, q} \rightarrow N_{j\mu l, r} \mid p_{\lambda k, q}, p_{\mu l, r} \in \mathcal{P}, y_{\lambda k}^{c_a} z_i^{c_a} = y_{\mu l}^{c_b} z_j^{c_b} = 1, c_a, c_b \in \mathcal{C}\}, \quad (2)$$

where an individual arc $A_{i\lambda k, q \rightarrow j\mu l, r} \in A'_{i \rightarrow j}$ represents the data flow between candidate nodes $N_{i\lambda k, q}$ and $N_{j\mu l, r}$.

An example of transforming a TG G into ETG G' is illustrated in Fig. 1. G consists of four tasks/nodes, with tasks N_1 , N_2 , N_3 , and N_4 requiring device capabilities c_2 , c_1 , c_0 , and c_3 , respectively (entry/exit tasks are depicted in green/yellow and intermediate tasks in blue). The underlying edge-hub-cloud CPS comprises two edge devices u_{e1} and u_{e2} with capabilities $\mathcal{C}_{e1} = \{c_0, c_1\}$ and $\mathcal{C}_{e2} = \{c_0, c_2\}$, respectively, a hub device u_{h1} with capabilities $\mathcal{C}_{h1} = \{c_0, c_3\}$,

and a cloud server u_{c1} with capability $C_{c1} = \{c_0\}$. For the execution of G , one processing core is reserved on each of the devices u_{e1} , u_{e2} , and u_{h1} , and two processing cores on u_{c1} . Devices u_{e1} and u_{e2} utilize u_{h1} to communicate with u_{c1} . Based on the capability requirements of each task, and the capabilities featured by each device, the possible allocations of the tasks on the processing cores result in the generation of eight candidate nodes in G' (one candidate node for N_1 , N_2 , and N_4 , and five for N_3). Each arc between two tasks in G is transformed into a set of arcs in G' connecting all pairs of candidate nodes of the two tasks. The arcs in G' that involve indirect communication between devices are shown in orange.

In the worst case of our transformation technique, where all devices feature all capabilities in $\mathcal{C}$, for γ processing cores the number of candidate nodes in ETG G' increases by γ times, and the number of arcs by γ^2 times, compared to those in TG G . For applications like the ones motivating this work [1]–[4], which do not typically have an excessive number of tasks, the optimality of our approach outweighs the moderate complexity introduced by the increase in task graph size. This is demonstrated experimentally in Section V-C.

1) *Candidate Node Parameters*: In addition to the binary parameters $y_{\lambda k}^{c_a}$ and $z_i^{c_a}$ defined in Section III-B, a candidate node $N_{i\lambda k.q} \in G'$ has the following parameters:

- D_i : output data size of task τ_i .
- M_i : main memory required by τ_i .
- S_i : storage required by τ_i .
- $\mathcal{Q}_i$: set of child tasks of τ_i .
- $L_{i\lambda k.q}$: execution time of τ_i on processing core $p_{\lambda k.q}$.
- $P_{i\lambda k.q}$: power required to execute τ_i on $p_{\lambda k.q}$.
- $E_{i\lambda k.q}$: energy required to execute τ_i on $p_{\lambda k.q}$, given by:

$$E_{i\lambda k.q} = L_{i\lambda k.q} P_{i\lambda k.q}. \quad (3)$$

2) *Arc Parameters*: An arc $A_{i\lambda k.q \rightarrow j\mu l.r} \in G'$ has the following parameters:

- $\sigma_{i\lambda k.q \rightarrow j\mu l.r}^{\nu m}$: binary parameter denoting whether $A_{i\lambda k.q \rightarrow j\mu l.r}$ involves indirect communication between devices $u_{\lambda k}$ and $u_{\mu l}$ through device $u_{\nu m}$:

$$\sigma_{i\lambda k.q \rightarrow j\mu l.r}^{\nu m} = \begin{cases} 1, & \text{if } \mathcal{I}_{\lambda k, \mu l} = \{u_{\nu m}\}, \\ 0, & \text{if } \mathcal{I}_{\lambda k, \mu l} = \emptyset. \end{cases} \quad (4)$$

- $CL_{i\lambda k.q \rightarrow j\mu l.r}$: time required to transfer the output data D_i of task τ_i that is allocated on $p_{\lambda k.q}$, to task τ_j that is allocated on $p_{\mu l.r}$:

$$CL_{i\lambda k.q \rightarrow j\mu l.r} = \begin{cases} \frac{D_i}{\theta_{\lambda k, \mu l}}, & \text{if } \sigma_{i\lambda k.q \rightarrow j\mu l.r}^{\nu m} = 0, \\ & (\lambda, k) \neq (\mu, l), \\ D_i \left(\frac{1}{\theta_{\lambda k, \nu m}} + \frac{1}{\theta_{\nu m, \mu l}} \right), & \text{if } \sigma_{i\lambda k.q \rightarrow j\mu l.r}^{\nu m} = 1, \\ 0, & \text{if } (\lambda, k) = (\mu, l). \end{cases} \quad (5)$$

- $CE_{i\lambda k.q \rightarrow j\mu l.r}$: energy required to transfer D_i from task τ_i (allocated on $p_{\lambda k.q}$) to task τ_j (allocated on $p_{\mu l.r}$):

$$CE_{i\lambda k.q \rightarrow j\mu l.r} = \begin{cases} D_i (\pi_{\lambda k, \mu l} + \rho_{\lambda k, \mu l}), & \text{if } \sigma_{i\lambda k.q \rightarrow j\mu l.r}^{\nu m} = 0, \\ & (\lambda, k) \neq (\mu, l), \\ D_i (\pi_{\lambda k, \nu m} + \rho_{\lambda k, \nu m} + \pi_{\nu m, \mu l} + \rho_{\nu m, \mu l}), & \text{if } \sigma_{i\lambda k.q \rightarrow j\mu l.r}^{\nu m} = 1, \\ 0, & \text{if } (\lambda, k) = (\mu, l). \end{cases} \quad (6)$$

D. Problem Formulation

We leverage the resulting ETG G' to formulate the scheduling problem as a continuous-time MILP model, as follows.

1) *Decision Variables*: We employ the following variables:

- $x_{i\lambda k.q}$: binary variable corresponding to a candidate node $N_{i\lambda k.q} \in G'$, such that $x_{i\lambda k.q} = 1$ if $N_{i\lambda k.q}$ is selected, and $x_{i\lambda k.q} = 0$ otherwise.
- $x_{i\lambda k.q \rightarrow j\mu l.r}$: binary variable corresponding to an arc $A_{i\lambda k.q \rightarrow j\mu l.r} \in G'$, such that $x_{i\lambda k.q \rightarrow j\mu l.r} = 1$ if $A_{i\lambda k.q \rightarrow j\mu l.r}$ is selected, and $x_{i\lambda k.q \rightarrow j\mu l.r} = 0$ otherwise.
- t_i : continuous variable denoting the start time of task τ_i , corresponding to the set $N'_i \in G'$.
- T : continuous variable denoting the completion time of the application.
- $x_{i,j}$: auxiliary binary variable denoting whether task τ_i will be executed before task τ_j ($x_{i,j} = 1$) or not ($x_{i,j} = 0$). It is used when both tasks are allocated on the same processing core and do not have a precedence relationship between them, and thus their order of execution should be determined to avoid any overlap.
- $x_n^{i\lambda k.q}$: auxiliary binary variable denoting whether task τ_i will be in execution on its allocated processing core $p_{\lambda k.q}$ at time e_n ($x_n^{i\lambda k.q} = 1$) or not ($x_n^{i\lambda k.q} = 0$). It is used to prevent exceeding the capability, memory, and storage capacity of a device during the execution of its assigned tasks (i.e., it is used to formulate time-dependent cumulative constraints). For this purpose, taking into account that the time interval in which a task τ_i will be in execution on $p_{\lambda k.q}$ is given by $[t_i, t_i + L_{i\lambda k.q})$, we consider a set comprising the start times of all tasks, i.e., $\mathcal{E} = \{e_n \mid e_n = t_i, N'_i \in \mathcal{N}'\}$. To preserve the linearity of the model, an additional related auxiliary binary variable $\hat{x}_n^{i\lambda k.q}$ is used.

2) *Objective & Constraints*: Our aim is to minimize the overall latency (i.e., the completion time) of the application:

$$\min T \quad (7)$$

subject to the following constraints.

a) *Candidate node selection constraints*: Only one candidate node per task should be selected in G' , i.e., each task should be allocated on only one processing core:

$$\sum_{N_{i\lambda k.q} \in N'_i} x_{i\lambda k.q} = 1, \forall N'_i \in \mathcal{N}'. \quad (8)$$

b) *Arc selection constraints*: If a parent and child candidate nodes are selected, their corresponding arc should be selected as well:

$$x_{i\lambda k.q \rightarrow j\mu l.r} \leq x_{i\lambda k.q}, \forall A_{i\lambda k.q \rightarrow j\mu l.r} \in \mathcal{A}', \quad (9)$$

$$x_{i\lambda k.q \rightarrow j\mu l.r} \leq x_{j\mu l.r}, \forall A_{i\lambda k.q \rightarrow j\mu l.r} \in \mathcal{A}', \quad (10)$$

$$x_{i\lambda k.q \rightarrow j\mu l.r} \geq x_{i\lambda k.q} + x_{j\mu l.r} - 1, \forall A_{i\lambda k.q \rightarrow j\mu l.r} \in \mathcal{A}'. \quad (11)$$

c) *Task precedence constraints:* The precedence relationships among the tasks should be preserved:

$$t_i + L_{i\lambda k.q} x_{i\lambda k.q} + CL_{i\lambda k.q \rightarrow j\mu l.r} x_{i\lambda k.q \rightarrow j\mu l.r} \leq t_j, \quad \forall A_{i\lambda k.q \rightarrow j\mu l.r} \in \mathcal{A}'. \quad (12)$$

d) *Application completion time & deadline constraints:* The completion time of the application should be equal to the completion time of its last task (13), and within the predefined deadline (14):

$$t_i + L_{i\lambda k.q} x_{i\lambda k.q} \leq T, \forall N_{i\lambda k.q} \in \mathcal{N}', \quad (13)$$

$$T \leq L_{\text{thr}}. \quad (14)$$

e) *Task non-overlapping constraints:* Two tasks without a precedence relationship between them, allocated on the same processing core, should not be executed at the same time, as each processing core can process only one task at a time:

$$t_i + L_{i\lambda k.q} x_{i\lambda k.q} \leq t_j + (3 - x_{i\lambda k.q} - x_{j\lambda k.q} - x_{i,j}) \Omega, \quad \forall N_{i\lambda k.q}, N_{j\lambda k.q} \in \mathcal{N}', i < j, \tau_i \notin \mathcal{Q}_j, \tau_j \notin \mathcal{Q}_i, \quad (15)$$

$$t_j + L_{j\lambda k.q} x_{j\lambda k.q} \leq t_i + (2 - x_{i\lambda k.q} - x_{j\lambda k.q} + x_{i,j}) \Omega, \quad \forall N_{i\lambda k.q}, N_{j\lambda k.q} \in \mathcal{N}', i < j, \tau_i \notin \mathcal{Q}_j, \tau_j \notin \mathcal{Q}_i. \quad (16)$$

f) *Task execution constraints:* The time instants $e_n \in \mathcal{E}$ at which each task will be in execution on its allocated processing core are determined by the following constraints:

$$x_n^{i\lambda k.q} \leq x_{i\lambda k.q}, \forall e_n \in \mathcal{E}, \forall N_{i\lambda k.q} \in \mathcal{N}', \quad (17)$$

$$t_i \leq e_n + (2 - x_{i\lambda k.q} - x_n^{i\lambda k.q}) \Omega, \forall e_n \in \mathcal{E}, \forall N_{i\lambda k.q} \in \mathcal{N}', \quad (18)$$

$$e_n + \epsilon \leq t_i + L_{i\lambda k.q} x_{i\lambda k.q} + (2 - x_{i\lambda k.q} - x_n^{i\lambda k.q}) \Omega, \quad \forall e_n \in \mathcal{E}, \forall N_{i\lambda k.q} \in \mathcal{N}', \quad (19)$$

$$e_n + \epsilon \leq t_i + (2 - x_{i\lambda k.q} + x_n^{i\lambda k.q} - \hat{x}_n^{i\lambda k.q}) \Omega, \quad \forall e_n \in \mathcal{E}, \forall N_{i\lambda k.q} \in \mathcal{N}', \quad (20)$$

$$t_i + L_{i\lambda k.q} x_{i\lambda k.q} \leq e_n + (1 - x_{i\lambda k.q} + x_n^{i\lambda k.q} + \hat{x}_n^{i\lambda k.q}) \Omega, \quad \forall e_n \in \mathcal{E}, \forall N_{i\lambda k.q} \in \mathcal{N}'. \quad (21)$$

g) *Device capability, memory & storage constraints:* At each time instant $e_n \in \mathcal{E}$, no more than one task among those being executed on the processing cores of a particular device, should require a specific specialized capability (22). Similarly, at each time instant $e_n \in \mathcal{E}$ the memory (23) and storage (24) budgets of each device should not be exceeded:

$$\sum_{N_{i\lambda k.q} \in \mathcal{N}'} x_n^{i\lambda k.q} y_{\lambda k}^{c_a} z_i^{c_a} \leq 1, \forall e_n \in \mathcal{E}, \forall u_{\lambda k} \in \mathcal{U}, \forall c_a \in \mathcal{C}, a > 0, \quad (22)$$

$$\sum_{N_{i\lambda k.q} \in \mathcal{N}'} M_i x_n^{i\lambda k.q} \leq M_{\lambda k}^{\text{bgt}}, \forall e_n \in \mathcal{E}, \forall u_{\lambda k} \in \mathcal{U}, \quad (23)$$

$$\sum_{N_{i\lambda k.q} \in \mathcal{N}'} S_i x_n^{i\lambda k.q} \leq S_{\lambda k}^{\text{bgt}}, \forall e_n \in \mathcal{E}, \forall u_{\lambda k} \in \mathcal{U}. \quad (24)$$

h) *Device energy constraints:* The energy budget of each device should not be exceeded for the execution of the application, considering both the computational and communication energy requirements of the tasks. Regarding communication energy, we consider all data transmitted from and received at each device (either directly or indirectly to/from another device), including the case where a device is used for the communication between other devices:

$$\begin{aligned} & \sum_{N_{i\lambda k.q} \in \mathcal{N}'} E_{i\lambda k.q} x_{i\lambda k.q} \\ & + \sum_{A_{i\lambda k.q \rightarrow j\mu l.r} \in \mathcal{A}'} D_i x_{i\lambda k.q \rightarrow j\mu l.r} (\pi_{\lambda k, \mu l} (1 - \sigma_{i\lambda k.q \rightarrow j\mu l.r}^{\nu m}) \\ & \quad + \pi_{\lambda k, \nu m} \sigma_{i\lambda k.q \rightarrow j\mu l.r}^{\nu m}) \\ & + \sum_{A_{j\mu l.r \rightarrow i\lambda k.q} \in \mathcal{A}'} D_j x_{j\mu l.r \rightarrow i\lambda k.q} (\rho_{\mu l, \lambda k} (1 - \sigma_{j\mu l.r \rightarrow i\lambda k.q}^{\nu m}) \\ & \quad + \rho_{\nu m, \lambda k} \sigma_{j\mu l.r \rightarrow i\lambda k.q}^{\nu m}) \\ & + \sum_{A_{i\mu l.q \rightarrow j\nu m.r} \in \mathcal{A}'} D_i x_{i\mu l.q \rightarrow j\nu m.r} (\rho_{\mu l, \lambda k} + \pi_{\lambda k, \nu m}) \sigma_{i\mu l.q \rightarrow j\nu m.r}^{\lambda k} \\ & \leq E_{\lambda k}^{\text{bgt}}, \forall u_{\lambda k} \in \mathcal{U}, (\lambda, k) \neq (\mu, l) \neq (\nu, m). \end{aligned} \quad (25)$$

i) *Binary & non-negativity constraints:* The binary nature (26)–(29) and non-negativity (30), (31) of the binary and continuous decision variables, respectively, should be ensured:

$$x_{i\lambda k.q} \in \{0, 1\}, \forall N_{i\lambda k.q} \in \mathcal{N}', \quad (26)$$

$$x_{i\lambda k.q \rightarrow j\mu l.r} \in \{0, 1\}, \forall A_{i\lambda k.q \rightarrow j\mu l.r} \in \mathcal{A}', \quad (27)$$

$$x_{i,j} \in \{0, 1\}, \forall N_{i\lambda k.q}, N_{j\lambda k.q} \in \mathcal{N}', i < j, \tau_i \notin \mathcal{Q}_j, \tau_j \notin \mathcal{Q}_i, \quad (28)$$

$$x_n^{i\lambda k.q}, \hat{x}_n^{i\lambda k.q} \in \{0, 1\}, \forall e_n \in \mathcal{E}, \forall N_{i\lambda k.q} \in \mathcal{N}', \quad (29)$$

$$t_i \geq 0, \forall N_i' \in \mathcal{N}', \quad (30)$$

$$T \geq 0. \quad (31)$$

It is noted that in (19) and (20), we utilize a positive tolerance constant ϵ , sufficiently smaller than the variables and parameters used in the model, to convert the constraints to non-strict inequalities, as strict inequalities are not supported in MILP. Furthermore, in (15), (16), (18)–(21), we employ a constant Ω , sufficiently larger than the worst-case completion time of any task of the application, to formulate the conditional aspects of the constraints in linear form. For example, (15) becomes meaningful (i.e., $t_i + L_{i\lambda k.q} x_{i\lambda k.q} \leq t_j$) only if tasks τ_i and τ_j are allocated on the same processing core $p_{\lambda k.q}$ (i.e., $x_{i\lambda k.q} = x_{j\lambda k.q} = 1$) and τ_i is executed before τ_j (i.e., $x_{i,j} = 1$). Otherwise, Ω remains on the right-hand side of (15), forcing it to be always true. Constraints (16), (18)–(21) are modeled in a similar fashion.

IV. HEFT EXTENSION

As showcased in Section II, our MILP approach is the first to optimally solve the examined problem in the considered CPS under the specific constraints. Thus, any modifications to other exact methods to account for the particular constraints would inevitably lead to a formulation identical to ours. Therefore, we chose to compare our approach with HEFT [8]–[10], which is one of the most frequently used and effective scheduling heuristics for workflow applications. HEFT involves two phases, (a) a task prioritization phase, where tasks are prioritized according to their upward rank (i.e., the longest distance to an exit task, in terms of latency), and (b) a processing core selection phase, where each task, in order of priority, is allocated to the processing core that can provide it with the minimum finish time, utilizing any schedule gaps.

As HEFT does not inherently support all the constraints of the proposed MILP approach, we extended it to ensure a meaningful and fair comparison. Specifically, we adapted HEFT to leverage the ETG G' of the application (rather than TG G), and enhanced it by incorporating into its second phase the deadline (14), capability (22), memory (23), storage (24), and energy (25) constraints considered in our method (the original version of HEFT [8] supports only the precedence (12) and non-overlapping (15), (16) constraints). As the enhanced version of HEFT (shown in Algorithm 1) takes as input the ETG G' , its first phase is converted into a candidate node prioritization phase (lines 1–5), and its second phase into a candidate node selection phase (lines 6–59).

In the first phase, the upward rank of a candidate node $N_{i\lambda k.q}$ (line 2) is calculated based on the upward rank of its child nodes [8]:

$$R_{i\lambda k.q} = L_{i\lambda k.q} + \max_{A_{i\lambda k.q \rightarrow j\mu l.r} \in \mathcal{A}'} \{CL_{i\lambda k.q \rightarrow j\mu l.r} + R_{j\mu l.r}\}. \quad (32)$$

Candidate nodes are prioritized based on their upward rank (line 5). In the second phase, for each unscheduled task τ_i (for which no candidate node has been selected) we examine each of its candidate nodes $N_{i\lambda k.q} \in \mathcal{N}'_i$ (in order of rank) to select the one that minimizes its finish time. In particular, we first check if selecting a candidate node $N_{i\lambda k.q}$ would exceed the energy budget of any device (lines 17–23). Moreover, we examine whether the memory or storage requirements of τ_i exceed the respective budgets of device $u_{\lambda k}$ (line 23). If any of these conditions hold, we skip $N_{i\lambda k.q}$ and continue with the next (according to its rank) candidate node of τ_i (line 24).

Otherwise, using $N_{i\lambda k.q}$ we determine the start time t_i that yields the earliest finish time $EFT_{i\lambda k.q}$ for τ_i on processing core $p_{\lambda k.q}$, so that its precedence constraints are satisfied (line 26), its execution does not overlap with other tasks already allocated on $p_{\lambda k.q}$, and the capability, memory, and storage capacities of device $u_{\lambda k}$ are not exceeded (lines 27–37). For the capacity constraints, we consider tasks already allocated on $u_{\lambda k}$ (but on different processing cores) and whose execution will overlap with τ_i (line 31). Subsequently, we select the candidate node that minimizes the earliest finish time of τ_i on any device/processing core, without exceeding the execution deadline L_{thr} (lines 44–54). If there is a task

Algorithm 1: Enhanced version of HEFT.

Input: ETG $G' = (\mathcal{N}', \mathcal{A}')$.
Output: $x_{i\lambda k.q} \forall N_{i\lambda k.q} \in \mathcal{N}'$, $x_{i\lambda k.q \rightarrow j\mu l.r} \forall A_{i\lambda k.q \rightarrow j\mu l.r} \in \mathcal{A}'$, and $t_i \forall N'_i \in G'$.

```

// Phase 1 - candidate node prioritization:
1 foreach candidate node  $N_{i\lambda k.q} \in \mathcal{N}'$  starting from exit task candidate nodes do
2   | Calculate upward rank  $R_{i\lambda k.q}$  using (32);
3 end
4  $\Lambda \leftarrow \mathcal{N}'$ ;
5 Sort list  $\Lambda$  by non-increasing order of  $R_{i\lambda k.q}$ ;
// Phase 2 - candidate node selection:
6  $isInfeasible \leftarrow 0$ ;
7  $\mathcal{N}'_{sel} \leftarrow \emptyset$ ;
8  $\mathcal{A}'_{sel} \leftarrow \emptyset$ ;
9 while  $\exists$  unscheduled tasks of candidate nodes in  $\Lambda$  do
10   $applCandNodeExists \leftarrow 0$ ;
11   $minEFT \leftarrow 0$ ;
12  Select first unscheduled task  $\tau_i$  (i.e.,  $N'_i \in G'$ ) according to  $\Lambda$ ;
13  foreach candidate node  $N_{i\lambda k.q} \in \mathcal{N}'_i$  do
14     $\mathcal{N}'_{temp} \leftarrow \mathcal{N}'_{sel} \cup \{N_{i\lambda k.q}\}$ ;
15     $\mathcal{A}'_{temp} \leftarrow \mathcal{A}'_{sel} \cup \{A_{j\mu l.r \rightarrow i\lambda k.q} \in \mathcal{A}' | N_{j\mu l.r} \in \mathcal{N}'_{sel}\}$ ;
16     $energyBgtExceeded \leftarrow 0$ ;
17    foreach device  $u_{\mu l} \in \mathcal{U}$  do
18      | if constraint (25) is violated for  $u_{\mu l}$  using  $\mathcal{N}'_{temp}, \mathcal{A}'_{temp}$  in place of
19        |  $\mathcal{N}', \mathcal{A}'$ , respectively then
20        |    $energyBgtExceeded \leftarrow 1$ ;
21        |   Break;
22    end
23    if  $energyBgtExceeded \vee M_i > M_{\lambda k}^{bgt} \vee S_i > S_{\lambda k}^{bgt}$  then
24      | Continue; //  $N_{i\lambda k.q}$  is not applicable
25    else
26       $t_i \leftarrow \max_{A_{j\mu l.r \rightarrow i\lambda k.q} \in \mathcal{A}'_{temp}} \{t_j + L_{j\mu l.r} + CL_{j\mu l.r \rightarrow i\lambda k.q}\}$ ;
27       $\Lambda_{\lambda k} \leftarrow \{N_{j\lambda k.r} \in \mathcal{N}'_{sel} | t_j + L_{j\lambda k.r} > t_i\}$ ;
28      Sort list  $\Lambda_{\lambda k}$  by non-decreasing order of  $t_j + L_{j\lambda k.r}$ ;
29      for  $n \leftarrow 1$  to  $|\Lambda_{\lambda k}|$  do
30        |  $N_{j\lambda k.r} \leftarrow \Lambda_{\lambda k}[n]$ ;
31        |  $\mathcal{N}'_{temp} \leftarrow \{N_{i\lambda k.q}\} \cup \{N_{h\lambda k.s} \in \Lambda_{\lambda k} | s \neq q, t_i <$ 
32        |    $(t_h + L_{h\lambda k.s}) \wedge (t_i + L_{i\lambda k.q}) > t_h\}$ ;
33        | if  $(r = q \wedge t_i < (t_j + L_{j\lambda k.r}) \wedge (t_i + L_{i\lambda k.q}) > t_j)$ 
34        |    $\vee \left( \sum_{N_{h\lambda k.s} \in \mathcal{N}'_{temp}} y_{\lambda k}^{ca} z_h^{ca} > 1 \wedge z_i^{ca} = 1 \wedge a > 0 \right)$ 
35        |    $\vee \sum_{N_{h\lambda k.s} \in \mathcal{N}'_{temp}} M_h > M_{\lambda k}^{bgt} \vee \sum_{N_{h\lambda k.s} \in \mathcal{N}'_{temp}} S_h > S_{\lambda k}^{bgt}$  then
36        |     |  $t_i \leftarrow t_j + L_{j\lambda k.r}$ ;
37        |   end
38        |    $n \leftarrow n + 1$ ;
39      end
40       $EFT_{i\lambda k.q} \leftarrow t_i + L_{i\lambda k.q}$ ;
41      if  $minEFT = 0 \vee EFT_{i\lambda k.q} < minEFT$  then
42        |  $minEFT \leftarrow EFT_{i\lambda k.q}$ ;
43      end
44       $applCandNodeExists \leftarrow 1$ ;
45    end
46  end
47  if  $applCandNodeExists \wedge minEFT \leq L_{thr}$  then
48    | Select candidate node  $N_{i\lambda k.q}$  that has  $EFT_{i\lambda k.q} = minEFT$ ;
49    |  $x_{i\lambda k.q} \leftarrow 1$ ;
50    | Mark task  $\tau_i$  as scheduled;
51    |  $\mathcal{N}'_{sel} \leftarrow \mathcal{N}'_{sel} \cup \{N_{i\lambda k.q}\}$ ;
52    | foreach  $A_{j\mu l.r \rightarrow i\lambda k.q} \in \mathcal{A}'$  do
53    |   | if  $N_{j\mu l.r} \in \mathcal{N}'_{sel}$  then
54    |     |  $x_{j\mu l.r \rightarrow i\lambda k.q} \leftarrow 1$ ;
55    |     |  $\mathcal{A}'_{sel} \leftarrow \mathcal{A}'_{sel} \cup \{A_{j\mu l.r \rightarrow i\lambda k.q}\}$ ;
56    |   end
57    | end
58  else
59    |  $isInfeasible \leftarrow 1$ ;
60    | Break;
61  end
62 return  $isInfeasible$ ;

```

for which no candidate node can satisfy the above constraints, then the problem is infeasible. Considering that for dense ETGs the number of arcs $|\mathcal{A}'|$ is proportional to $|\mathcal{N}'|^2$ [8], the worst-case time complexity of the first phase is dominated by the operations performed in lines 1–3. Thus, its worst-case time complexity is $O(|\mathcal{N}'|^3)$. Similarly, the worst-case time complexity of the second phase is dominated by the

TABLE II
DEVICE CAPABILITIES

#	Description
0	Basic computational capability
1	Thermal camera
2	LiDAR sensor
3	Multispectral camera
4	High-precision GNSS module
5	Custom payload release mechanism
6	Specialized software module (e.g., for UAV coordination)
7	Integrated display unit
8	Specialized hardware accelerator (e.g., a GPU)
9	High-availability storage

operations in lines 17–22, i.e., $O(|U||N'|^3)$. Evidently, the time complexity of the second phase is greater than that of the first phase. Hence, the overall worst-case time complexity of extended HEFT is $O(|U||N'|^3)$. It is noted that the time complexity of the proposed optimal MILP approach depends not only on the size of the problem, but also on the utilized solver. Commercial solvers (e.g., Gurobi [24]) typically employ proprietary algorithms whose implementation details are not publicly available, and thus their time complexity cannot be easily derived [13]. With regard to problem size, as mentioned in Section III-C, in the worst case the number of nodes and arcs in the resulting ETG G' increases linearly and quadratically, respectively, compared to TG G .

V. EXPERIMENTAL EVALUATION

We evaluated and compared the proposed MILP approach against the enhanced version of HEFT, considering a real-world workflow under various system configurations. To further validate and examine the scalability of our technique to applications of different sizes, we developed and used synthetic workflows with appropriate parameters.

A. Experimental Setup

We investigated an edge-hub-cloud CPS with multiple edge devices in six different configurations (C1–C6). In each configuration, we considered varying numbers and types of edge devices that could communicate with each other and a hub device, which in turn could communicate with a cloud server. The devices were based on typical real-world counterparts, featuring heterogeneous multicore processors, and various memory, storage, and energy capacities. All devices featured different sensing/actuating or other specialized capabilities based on the real-world use case. These capabilities are indicated by integers in the range $[0, 9]$, as shown in Table II.

The number of reserved processing cores $\gamma_{\lambda k}$ and the budgets $M_{\lambda k}^{\text{bgt}}$, $S_{\lambda k}^{\text{bgt}}$, and $E_{\lambda k}^{\text{bgt}}$ were a subset of the respective ones featured by each device, as computational resources in such use cases are typically limited and shared among different applications. Table III shows the system devices and their specifications, the considered budgets, as well as the capabilities of each device (per configuration). In addition to the hub device (Mi Notebook Pro) and the cloud server (HPE DL580 Gen10), each configuration included two (C1, C2), three (C3, C4), or four (C5, C6) edge devices selected from Raspberry Pi 3, Odroid XU4, Jetson TX2, and Jetson Xavier NX. Each edge device was considered to be attached to a UAV with different sensing/actuating capabilities. For example, configuration C1

included Raspberry Pi 3 with capabilities $\{0, 1, 3\}$, Jetson Xavier NX with capabilities $\{0, 2, 4, 5\}$, Mi Notebook Pro with capabilities $\{0, 6, 7\}$, and HPE DL580 Gen10 with capabilities $\{0, 8, 9\}$, as shown in the respective column of Table III. Table IV includes the ranges of the bandwidth and energy parameters for the communication channels between each pair of devices, which were based on real-world measurements [25], [26]. The model of the CPS under study is illustrated in Fig. 2. We implemented both the proposed MILP method and the extended version of HEFT in C++. In the MILP approach, the formulated problem was solved using Gurobi Optimizer 11 [24], on a server equipped with an Intel Xeon Gold 6240 processor @ 2.6 GHz and 400 GiB of RAM.

B. Real-World Workflow

1) *Overview*: We considered a relevant real-world workflow for the autonomous UAV-based inspection of power transmission towers and lines, based on [4]. It consists of 16 tasks, as shown in Table V and Fig. 3. Entry and exit tasks are depicted in green and yellow, respectively, while intermediate tasks are shown in blue. The required capabilities of the tasks (as described in Table II) are indicated by the red integers (0–9). In this use case, multiple UAVs with attached edge devices collaborate to capture multispectral images to detect power transmission lines and vegetation encroachment (tasks N_1 – N_4), perform LiDAR scans and integrate them with high-precision global navigation satellite system (GNSS) data to detect power towers and structural integrity problems (tasks N_5 – N_8), and capture thermal images to detect overheating components such as insulators (tasks N_9 – N_{11}).

The diverse data from all UAVs are fused to create a comprehensive visual representation of the infrastructure (task N_{12}). This is a computationally intensive process necessitating a specialized hardware accelerator, such as a high-performance GPU provided by a cloud server [27]. The fused data are saved on high-availability storage in the cloud to ensure enhanced reliability and facilitate remote access (task N_{13}). In addition, the fused data are used for tag deployment path planning and coordination, a process requiring a specialized software module for UAV control, typically installed on the hub device to improve latency (task N_{14}). Finally, the output is displayed on the hub device (task N_{15}), while a UAV starts deploying location-transmitting tags at the identified problematic sections of the infrastructure (to facilitate the ground crew in locating them) using a custom payload release mechanism (task N_{16}).

For the execution of the application, we considered two, three, or four edge devices (each attached to a different UAV with varied capabilities), as well as a hub device and a cloud server, based on the configurations C1–C6 in Table III. We transformed the TG of the application into the corresponding ETGs, one for each system configuration. The number of ETG candidate nodes/arcs in each case is shown in Table VI. The ETG parameters D_i , M_i , S_i , $L_{i\lambda k.q}$, and $P_{i\lambda k.q}$ were determined using profiling and power monitoring tools (perf and Powertop) [28] or through relevant benchmarks (Phoronix Test Suite) [29]. The ranges of these parameters are listed in Table VII. $E_{i\lambda k.q}$, $CL_{i\lambda k.q \rightarrow j\mu l.r}$, and $CE_{i\lambda k.q \rightarrow j\mu l.r}$ were

TABLE III
SYSTEM DEVICES & CONFIGURATIONS

$u_{\lambda k}$	Device	Specifications				Perf. Ratio	Budgets			Capabilities per Configuration ⁴						
		Processor	Memory (GiB)	Storage (GiB)	Battery ³ (Wh)		$\gamma_{\lambda k}$	$M_{\lambda k}^{\text{bgt}}$ (GiB)	$S_{\lambda k}^{\text{bgt}}$ (GiB)	$E_{\lambda k}^{\text{bgt}}$ (Wh)	C1	C2	C3	C4	C5	C6
u_{e1}	Raspberry Pi 3	Cortex-A53 @ 1.4 GHz	1	16	33.3	1.00	2	0.95	1.0	1	{0, 1, 3}	$\emptyset$	{0, 3}	{0, 3}	{0, 1}	{0, 1}
u_{e2}	Odroid XU4	Cortex-A7 & Cortex-A15 ¹ @ 2.0 GHz	2	16	33.3	1.20	2	1.00	1.5	1	$\emptyset$	{0, 1, 3}	{0, 5}	{0, 5}	{0, 3}	{0, 3}
u_{e3}	Jetson TX2	NVIDIA Denver2 & Cortex-A57 ² @ 2.0 GHz	8	32	33.3	2.80	2	2.00	2.0	1	$\emptyset$	{0, 2, 4, 5}	$\emptyset$	{0, 1, 2, 4}	{0, 5}	{0, 2, 4}
u_{e4}	Jetson Xavier NX	NVIDIA Carmel ARMv8.2 @ 1.4 GHz	8	32	33.3	5.74	2	2.00	2.5	1	{0, 2, 4, 5}	$\emptyset$	{0, 1, 2, 4}	$\emptyset$	{0, 2, 4}	{0, 5}
u_{h1}	Mi Notebook Pro	Intel i5 8250U @ 1.6 GHz	8	512	60.0	15.23	4	3.00	5.0	2	{0, 6, 7}	{0, 6, 7}	{0, 6, 7}	{0, 6, 7}	{0, 6, 7}	{0, 6, 7}
u_{c1}	HPE DL580 Gen10	Intel Xeon Gold 6240 @ 2.6 GHz	400	10240	—	21.70	6	4.00	10.0	10	{0, 8, 9}	{0, 8, 9}	{0, 8, 9}	{0, 8, 9}	{0, 8, 9}	{0, 8, 9}

^{1,2}Without loss of generality, the reserved processing cores are considered to be located on Cortex-A15 and Cortex-A57, respectively.

³A compatible external battery (TalentCell YB1203000-USB) is considered for the edge devices. ⁴ $\emptyset$ denotes that a device is not included in a configuration.



Fig. 2. System model.

TABLE IV
COMMUNICATION CHANNELS

Comm. Channel	$\theta_{\lambda k, \mu l}$ (Mbit/s)	$\pi_{\lambda k, \mu l}$ (μ J/bit)	$\rho_{\lambda k, \mu l}$ (μ J/bit)
$u_{ek} \leftrightarrow u_{el}$	[6, 9]	[0.6, 1.0]	[0.4, 0.6]
$u_{ek} \rightarrow u_{h1}$	[9, 13]	[0.8, 1.2]	[0.6, 0.8]
$u_{h1} \rightarrow u_{el}$	[7, 10]	[0.7, 1.1]	[0.5, 0.7]
$u_{h1} \rightarrow u_{c1}$	[10, 15]	[1.8, 2.7]	[0.8, 1.2]
$u_{c1} \rightarrow u_{h1}$	[16, 24]	[2.0, 3.0]	[1.0, 1.5]

TABLE VI
REAL-WORLD WORKFLOW ETGS

Config.	#Nodes/Arcs	#Variables/ Constraints	Solver Runtime (s)
C1	128/1208	5705/16353	6.68
C2	128/1208	5705/16353	8.71
C3	142/1496	6455/18736	11.19
C4	142/1496	6455/18736	13.05
C5	156/1816	7237/21243	14.80
C6	156/1816	7237/21243	20.62

TABLE VII
REAL-WORLD WORKFLOW
ETG PARAMETER RANGES

Param.	Range
D_i	[3.22, 151.27] Mbit
M_i	[103.98, 3800.17] Mbit
S_i	[246.57, 3766.42] Mbit
$L_{i\lambda k, q}$	[2.57, 12648.38] ms
$P_{i\lambda k, q}$	[0.30, 23.70] W

TABLE V
REAL-WORLD WORKFLOW TASKS

Task	Description
N_1	Capture multispectral image
N_2	Multispectral image preprocessing
N_3	Detect power transmission lines
N_4	Detect vegetation encroachment
N_5	Perform LiDAR scan
N_6	Data preprocessing/integrate GNSS data
N_7	Detect power towers
N_8	Detect structural integrity problems
N_9	Capture thermal image
N_{10}	Thermal image preprocessing
N_{11}	Detect overheating components
N_{12}	Multi-source data fusion
N_{13}	Save fused data on high-availability storage
N_{14}	Path planning/coordinate tag deployment
N_{15}	Display final output
N_{16}	Deploy tags at problematic sections

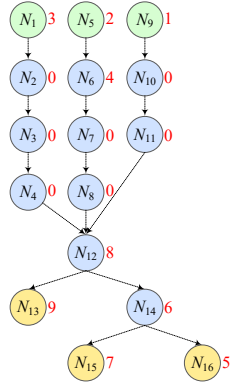


Fig. 3. Real-world workflow TG.

calculated using (3), (5), and (6), respectively, whereas Q_i and $\sigma_{i\lambda k, q \rightarrow j\mu l, r}^{vm}$ were derived from the respective TG and ETG structures. For each ETG, we set the deadline L_{thr} to be 1.5 times greater than its critical path (considering the computational and communication latency of the candidate nodes and arcs, respectively), as this was a realistic, but also challenging scenario [16].

2) *Evaluation Results:* We used the real-world workflow to compare our MILP approach with the extended version of HEFT. Our evaluation was primarily focused on overall latency, which is our problem objective (7). In addition, we considered the overall energy consumption as a secondary metric. It was calculated based on the resulting schedule in each case, by summing the left-hand side of (25) for all system devices. Fig. 4a showcases the results of the comparative evaluation between MILP and HEFT for each system configuration, with respect to overall latency. It can be observed that MILP consistently outperformed HEFT in all of the examined scenarios. This superiority is further highlighted in Fig. 4b, which demonstrates the latency improvement achieved by MILP over HEFT, under each configuration. Notably, the proposed MILP technique provided an average latency decrease of 13.54%, across all cases.

Figs. 4a and 4b also show that the overall latency yielded by

each method, as well as the latency improvement attained by MILP, depended on the employed system configuration. In particular, the latency decrease presented significant fluctuations, ranging from 8.98% to 18.22% across all configurations. This indicates that in the examined problem, the number, type, and sensing/actuating capabilities of edge devices play a decisive role in the resulting overall latency. The poor performance of HEFT was due to its inherent limitation in making sequential scheduling decisions. In contrast to MILP, HEFT schedules one task at a time, without assessing the entire solution space. Consequently, in HEFT a decision that is optimal at a specific step does not necessarily lead to a globally optimal schedule.

Regarding the overall energy consumption, which served as a secondary metric, Fig. 4c shows that MILP yielded a lower consumption than HEFT in all scenarios, even though the minimization of energy was not the objective of either technique. Specifically, MILP provided average energy savings of 14.88% over HEFT. The time required by the Gurobi solver to return a solution for the proposed MILP approach ranged between 6.68 and 20.62 s, as shown in Table VI. Table VI also demonstrates the number of generated variables and constraints in each case. Considering the pre-programmed nature of the examined workflow, which allows for its scheduling to be performed offline, the solver runtime is short and practical.

C. Scalability Analysis

1) *Generation of Synthetic Workflows:* To investigate the scalability of the proposed MILP approach to workflows of various sizes, we generated random TGs using the generator in [30]. Specifically, based on the real-world workflow in Section V-B, and considering that similar cyber-physical applications for the examined system architecture typically have a coarse-grained structure with a small-to-moderate number of tasks [1]–[4], we generated 25 random TGs grouped into five sets of different sizes. Each set comprised five TGs with 10, 20, 30, 40, or 50 tasks/nodes, and an average in/out degree (incoming/outgoing arcs per node) of 1.7/1.7. We randomly assigned specialized capabilities to the tasks of each TG, considering an 80% probability for the entry and exit tasks (as they

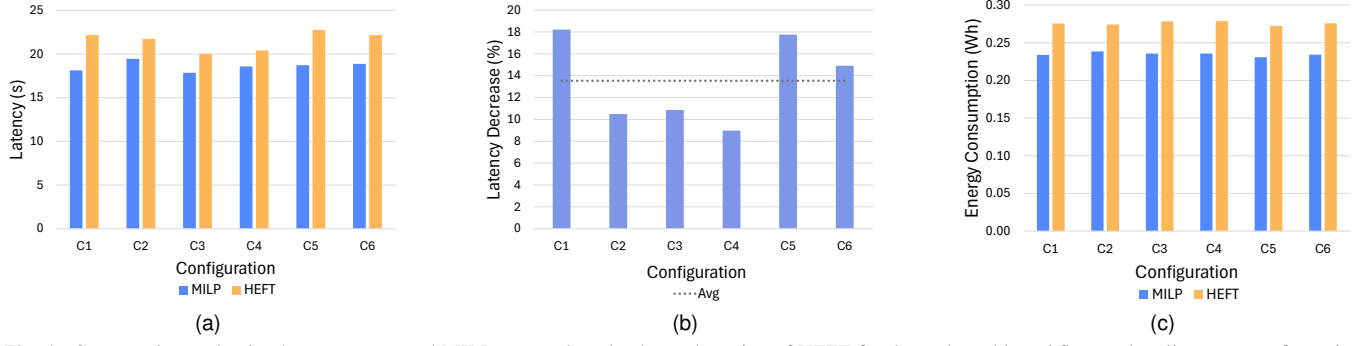


Fig. 4. Comparative evaluation between proposed MILP approach and enhanced version of HEFT for the real-world workflow under all system configurations.

TABLE VIII
SYNTHETIC WORKFLOW TGS & ETGS

TG Size (#Nodes)	TG Avg. #Arcs	ETG Avg. #Nodes/Arcs	Avg. #Variables/Constraints	Avg. Solver Runtime (min)
10	17	131/2871	5729/18846	0.20
20	36	240/6275	16553/52108	3.80
30	45	388/8464	33989/102391	31.78
40	65	492/11455	52916/157228	43.65
50	89	621/16917	82169/243625	99.24

usually require a specific sensor or actuator, respectively), and a 20% probability for the intermediate tasks. The remaining tasks were assigned the basic computational capability.

We subsequently transformed the generated TGs into the corresponding ETGs, based on configuration C6 (shown in Table III). We selected the particular configuration as it was among the most complex ones, encompassing four edge devices. The average number of nodes/arcs in each TG and ETG is shown in Table VIII. For the ETG candidate node parameters D_i , M_i , and S_i (which are device-independent) we randomly assigned values from the ones obtained in the real-world use case. For $L_{i\lambda k.q}$ and $P_{i\lambda k.q}$ (which are device/core-dependent), we first assigned values from those measured on Raspberry Pi 3 (used as a reference device) in the real-world scenario. Regarding the other devices, $L_{i\lambda k.q}$ and $P_{i\lambda k.q}$ were calculated based on the performance ratio of each device with respect to the reference device. The performance ratio (shown in Table III) was defined by comparing the performance score of each device to that of the reference device. The performance scores were obtained by running the Phoronix Test Suite benchmarks [29] on all devices (including the reference device). The remaining ETG candidate node and arc parameters were determined as described in Section V-B1. Our synthetic workflow datasets are openly accessible at [31].

2) *Scalability Results*: The scalability of the proposed MILP method was evaluated experimentally using the 25 synthetic workflows we generated. Fig. 5 demonstrates the latency improvement attained by our MILP approach over the enhanced version of HEFT, as the size of the workflows increased. Notably, MILP achieved a significant latency decrease for all workflows of all TG sizes. Specifically, MILP provided a latency improvement over HEFT ranging from 16.56% to 48.72%, with a mean and median of 33.03% and 33.6%, respectively, across all TG sizes. The latency decrease presented the highest variability for TGs with 10 nodes, due to the smaller number of tasks requiring specific capabilities.

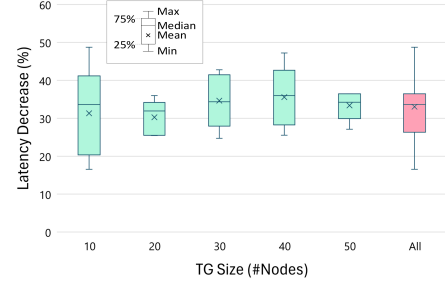


Fig. 5. Latency decrease achieved by proposed MILP approach over extended version of HEFT under increasing TG size. Box plot in red shows overall distribution of latency decrease across all TG sizes.

This resulted in ETGs with more diverse structures, compared to the other cases.

Fig. 6 shows the time required by the Gurobi solver to provide a solution for the proposed MILP technique as the TG size increased. The average solver execution time is also reported in Table VIII, along with the average number of variables/constraints in each case. The solver runtime ranged from 3 to 20s for 10-node TGs, 1 to 6 min for 20-node TGs, 5 to 72 min for 30-node TGs, 12 to 76 min for 40-node TGs, and 50 to 124 min for 50-node TGs. For reference, the runtime of the extended version of HEFT ranged from 71 to 4753 ms. Given the NP-hard nature of the examined problem [15], the optimality of the proposed MILP approach in terms of latency, and considering that this is an offline scheduling method where the primary objective is the minimization of workflow execution time (rather than the solver runtime), the time required by Gurobi is reasonable, showcasing the scalability of our technique as the problem size increased. The practicality of the proposed approach is further highlighted by the coarse-grained structure of relevant cyber-physical applications, which typically comprise 10–20 tasks, as demonstrated by the real-world workflow examined in Section V-B. For these TG sizes, the solver provided the optimal solution in a short time frame, ranging from 3 s to 6 min.

VI. CONCLUSION

We proposed an offline approach to optimally schedule a workflow application in an edge-hub-cloud CPS with heterogeneous multicore processors and various sensing, actuating, or other specialized capabilities. Our method utilizes a continuous-time MILP formulation to minimize the over-

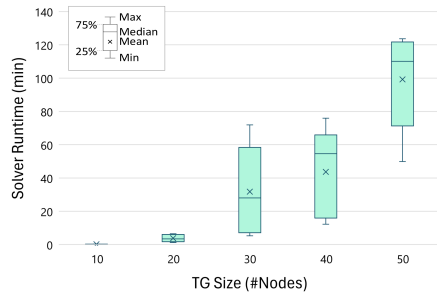


Fig. 6. Solver runtime for proposed MILP approach under increasing TG size.

all latency. It comprehensively considers multiple constraints often ignored by existing scheduling approaches, both exact and heuristic. Specifically, it addresses the memory, storage, and energy limitations of the devices, the heterogeneity and multicore architecture of the processors, the distinct device capabilities, the execution deadline, as well as the computational and communication latency and energy requirements of the tasks. Using a relevant real-world workflow, we compared our technique to the well-established HEFT scheduling heuristic, under different system configurations. In order for the comparison to be meaningful and fair, we extended HEFT by incorporating the deadline, capability, memory, storage, and energy constraints considered in our approach. Furthermore, we investigated the scalability of our method using representative synthetic workflows of various sizes and appropriate parameters. The experimental results revealed that the proposed technique consistently outperformed HEFT, yielding an average latency decrease of 13.54% in the real-world use case. Moreover, they demonstrated its scalability, as it provided an average latency improvement of 33.03% over HEFT for the synthetic workflows, within a reasonable time frame.

ACKNOWLEDGMENTS

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 739551 (KIOS CoE) and from the Government of the Republic of Cyprus through the Cyprus Deputy Ministry of Research, Innovation and Digital Policy.

REFERENCES

- [1] Q. Zheng, Q. Tang, Z. L. Wang, and Z. Li, "Self-powered cardiovascular electronic devices and systems," *Nat. Rev. Cardiol.*, vol. 18, no. 1, pp. 7–21, 2021.
- [2] R. Alam, J. Dugan, N. Homdee, N. Gandhi, B. Ghaemmaghami, H. Meda, A. Bankole, M. Anderson, J. Gong, T. Smith-Jackson, and J. Lach, "BESI: Reliable and heterogeneous sensing and intervention for in-home health applications," in *Proc. IEEE/ACM Int. Conf. Connect. Health Appl. Syst. Eng. Technol. (CHASE)*, 2017, pp. 147–156.
- [3] Z. Kashino, G. Nejat, and B. Benhabib, "Multi-UAV based autonomous wilderness search and rescue using target iso-probability curves," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, 2019, pp. 636–643.
- [4] A. Savva, A. Zacharia, R. Makrigiorgis, A. Anastasiou, C. Kyrkou, P. Kolios, C. Panayiotou, and T. Theodoridis, "ICARUS: Automatic autonomous power infrastructure inspection with UAVs," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, 2021, pp. 918–926.
- [5] L. Liu, S. Yao, X. Fu, H. Shao, R. Tabish, S. Yu, A. Bansal, H. Yun, L. Sha, and T. Abdelzaher, "Real-time task scheduling for machine perception in intelligent cyber-physical systems," *IEEE Trans. Comput.*, vol. 71, no. 8, pp. 1770–1783, 2022.

- [6] H. Tang, C. Li, J. Bai, J. Tang, and Y. Luo, "Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud-edge environment," *Comput. Commun.*, vol. 134, pp. 70–82, 2019.
- [7] C. A. Floudas and X. Lin, "Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications," *Ann. Oper. Res.*, vol. 139, no. 1, pp. 131–162, 2005.
- [8] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, 2002.
- [9] C. Kühbacher, C. Mellwig, F. Haas, and T. Ungerer, "A functional programming model for embedded dataflow applications," in *Proc. IEEE Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, 2019, pp. 646–651.
- [10] S. Aldegheri, N. Bombieri, and H. Patel, "On the task mapping and scheduling for DAG-based embedded vision applications on heterogeneous multi-/many-core architectures," in *Proc. Des. Autom. Test Eur. Conf. Exhib. (DATE)*, 2020, pp. 1003–1006.
- [11] T. A. Genez, L. F. Bittencourt, and E. R. Madeira, "Time-discretization for speeding-up scheduling of deadline-constrained workflows in clouds," *Future Gener. Comput. Syst.*, vol. 107, pp. 1116–1129, 2020.
- [12] L. Mo, Q. Zhou, A. Kritikakou, and J. Liu, "Energy efficient, real-time and reliable task deployment on NoC-based multicores with DVFS," in *Proc. Des. Autom. Test Eur. Conf. Exhib. (DATE)*, 2022, pp. 1347–1352.
- [13] L. Mo, Q. Zhou, A. Kritikakou, and X. Cao, "Energy optimized task mapping for reliable and real-time networked systems," *ACM Trans. Sen. Netw.*, vol. 19, no. 4, pp. 1–26, 2023.
- [14] H.-Y. Wu and C.-R. Lee, "Energy efficient scheduling for heterogeneous fog computing architectures," in *Proc. IEEE Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, 2018, pp. 555–560.
- [15] G. v. Zengen, J. Yu, and L. C. Wolf, "Adaptive real-time scheduling for cooperative cyber-physical systems," in *Proc. IEEE Conf. Ind. Cyberphys. Syst. (ICPS)*, 2020, pp. 79–86.
- [16] Y. Bai, Y. Huang, G. Xie, R. Li, and W. Chang, "ASDYS: Dynamic scheduling using active strategies for multifunctional mixed-criticality cyber-physical systems," *IEEE Trans. Ind. Inf.*, vol. 17, no. 8, pp. 5175–5184, 2021.
- [17] R. Alsurdeh, R. N. Calheiros, K. M. Matawie, and B. Javadi, "Hybrid workflow scheduling on edge cloud computing systems," *IEEE Access*, vol. 9, pp. 134 783–134 799, 2021.
- [18] G. L. Stavrinides and H. D. Karatza, "A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments," *Multimed. Tools Appl.*, vol. 78, no. 17, pp. 24 639–24 655, 2019.
- [19] A. Kouloumpiris, T. Theodoridis, and M. K. Michael, "Metis: Optimal task allocation framework for the edge/hub/cloud paradigm," in *Proc. Int. Conf. Omni-Layer Intell. Syst. (COINS)*, 2019, pp. 128–133.
- [20] —, "Cost-effective time-redundancy based optimal task allocation for the edge-hub-cloud systems," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, 2020, pp. 368–373.
- [21] A. Kouloumpiris, M. K. Michael, and T. Theodoridis, "Reliability-aware task allocation latency optimization in edge computing," in *Proc. IEEE Int. Symp. On-Line Test. Robust Syst. Des. (IOLTS)*, 2019, pp. 200–203.
- [22] A. Kouloumpiris, G. L. Stavrinides, M. K. Michael, and T. Theodoridis, "An optimization framework for task allocation in the edge/hub/cloud paradigm," *Future Gener. Comput. Syst.*, vol. 155, pp. 354–366, 2024.
- [23] H. Baek, J. Kwak, and J. Lee, "Non-preemptive real-time multiprocessor scheduling beyond work-conserving," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2020, pp. 102–114.
- [24] Gurobi Optimization, <https://www.gurobi.com>, acc. Nov. 3, 2023.
- [25] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. Int. Conf. Mob. Syst. Appl. Serv. (MobiSys)*, 2012, pp. 225–238.
- [26] V. Papić, P. Šolić, A. Milan, S. Gotovac, and M. Polić, "High-resolution image transmission from UAV to ground station for search and rescue missions planning," *Appl. Sci.*, vol. 11, no. 5, pp. 1–19, 2021.
- [27] H. Gao, X. Zhu, Q. Guan, X. Yang, Y. Yao, W. Zeng, and X. Peng, "cuFSDAF: An enhanced flexible spatiotemporal data fusion algorithm parallelized using graphics processing units," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–16, 2022.
- [28] Powertop, <https://github.com/fenrus75/powertop>, acc. Nov. 3, 2023.
- [29] Open Benchmarking, <https://openbenchmarking.org>, acc. Nov. 3, 2023.
- [30] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proc. Int. Workshop Hardw./Softw. Codesign*, 1998, pp. 97–101.
- [31] A. Kouloumpiris, G. L. Stavrinides, M. K. Michael, and T. Theodoridis, "Datasets of synthetic workflows for cyber-physical edge-hub-cloud systems." [Online]. Available: <https://doi.org/10.5281/zenodo.11031941>