HAL
open science

Energy Optimized Task Mapping for Reliable and Real-Time Networked Systems

Lei Mo, Qi Zhou, Angeliki Kritikakou, Xianghui Cao

To cite this version:

Lei Mo, Qi Zhou, Angeliki Kritikakou, Xianghui Cao. Energy Optimized Task Mapping for Reliable and Real-Time Networked Systems. ACM Transactions on Sensor Networks, 2023, pp.1-24. 10.1145/3584985 . hal-03999363

HAL Id: hal-03999363

https://hal.science/hal-03999363

Submitted on 21 Feb 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Energy Optimized Task Mapping for Reliable and Real-Time Networked Systems

LEI MO, School of Automation, Southeast University, China
QI ZHOU, School of Cyber Science and Engineering, Southeast University, China
ANGELIKI KRITIKAKOU, University of Rennes, INRIA, IRISA, CNRS, France
XIANGHUI CAO, School of Automation, Southeast University, China

Energy efficiency, real-time response, and data transmission reliability are important objectives during networked systems design. This paper aims to develop an efficient task mapping scheme to balance these important but conflicting objectives. To achieve this goal, tasks are triplicated to enhance reliability and mapped on the wireless nodes of the networked systems with Dynamic Voltage and Frequency Scaling (DVFS) capabilities to reduce energy consumption while still meeting real-time constraints. Our contributions include the mathematical formulation of this task mapping problem as mixed-integer programming that balances node energy consumption, enhancing data reliability, under real-time and energy constraints. Compared with the State-of-the-Art (SoA), a joint-design problem is considered in this paper, where DVFS, task triplication, task allocation, and task scheduling are optimized concurrently. To find the optimal solution, the original problem is linearized, and a decomposition-based method is proposed. The optimality of the proposed method is proved rigorously. Furthermore, a heuristic based on the greedy algorithm is designed to reduce the computation time. The proposed methods are evaluated and compared through a series of simulations. The results show that the proposed triplication-based task mapping method on average achieves 24.84% runtime reduction and 28.62% energy saving compared to the SoA methods.

Additional Key Words and Phrases: Networked Systems, Task Triplication and Mapping, DVFS, Problem Linearization and Decomposition

1 INTRODUCTION

Networked systems, e.g., Wireless Sensor and Actuator Networks (WSAN) or Cyber-Physical Systems (CPS), monitor and control the physical world using wireless nodes, equipped with sensors, controllers, and actuators, supporting several functions of system applications [20, 30]. From the hardware perspective, wireless nodes are usually energy-constrained, especially when they are supplied by batteries. To reduce energy consumption, modern hardware platforms are enhanced with Dynamic Voltage and Frequency Scaling (DVFS) [8, 21], able to adjust the supply voltage and the clock frequency of the processors, where task execution takes place. From the software perspective, system applications usually consist of several dependent tasks that exchange data. In-network processing is typically used to reduce network traffic [35]. Following the "Fog/Edge-computing" model [1, 16, 27], instead of collecting and sending all data to a remote Base Station (BS), a part of the data processing is done on the wireless nodes, reducing the amount of data sent to the BS. Hence, sensing, processing, and controlling tasks, and data transmissions should be executed in a real-time and energy-efficient manner, meeting the time and the energy constraints of the system application, in order to improve the overall system performance (e.g., enhancing federated learning in mobile edge computing) [40, 43].

Meanwhile, wireless nodes have become susceptible to attacks [39], and data transmissions should be reliable. During a False Data Injection (FDI) attack [28], an attacker compromises the data to obtain manipulated data that can bypass the basic "faulty data" detection mechanisms. An FDI attack can be implemented by compromising physical sensors and the sensor communication network [33]. Such attacks may lead to significant costs due to unplanned failures or loss of human lives in safety-critical applications, e.g., in predictive maintenance systems used in industry 4.0 the attack can propagate from the sensor to the machine learning part and fool the system by

predicting a delayed asset failure or maintenance interval [33]. Common countermeasure methods for FDI are based on contingency analysis [32]. To analyze the contingencies, state estimation (e.g., Kalman filter) [9] is usually used to model the system's dynamics. Thus, the estimation errors are influenced by the accuracy of the system model and the length of the historical data (i.e., time window size). Considering a system is executed under real-time constraints, potentially only parts of the data can be analyzed within the available time. An alternative to the aforementioned methods is to apply task replication. By replicating tasks, data reliability is enhanced, while knowledge of the system model is not required. Executing task replicas on different wireless nodes provides redundant information on the receiver node, which can correct any information obtained by a compromised sensor or communication network due to the FDI.

Overall, *energy efficiency*, *real-time application execution* and *reliable transmission* are crucial goals in the networked systems [25], and they are significantly affected by way of deploying tasks over the wireless nodes. The way allocating tasks affects the number of required data transmissions. The fewer data is exchanged among the nodes, the lower the energy consumption and time for transmissions, and the probability of the data being attacked during the transmission. However, not all tasks can be assigned on the same node, due to task requirements for specific sensors or actuators, and the application's real-time constraint. Task replication can occur when tasks are not assigned on the same nodes to protect the data from attacks. However, task replication also affects energy consumption and real-time execution. Furthermore, an asymmetric task allocation will deplete the energy budget of some nodes, leading to network connectivity issues. Therefore, task replication and deployment should be jointly addressed to achieve energy balance, under real-time and energy constraints.

Mapping tasks on a single platform, with one or several processors, are the well-known problems, e.g., task mapping to minimize energy consumption considering DVFS [8, 21, 22] and to improve system reliability using task replication [17, 42, 46]. However, works to address task deployment over different platforms, e.g., wireless nodes, considering task mapping, DVFS, reliability, and real-time constraints are rare. On the one hand, networked approaches focusing on task mapping [4, 12, 24, 35, 41] usually assume that the task results are reliable, as long as the task execution process is finished and the task data is transmitted to the destination, before the task deadline. On the other hand, networked approaches focusing on reliability use false data detection and correction, which are based on state estimation [9, 28, 31, 44]. Compared with the State-of-the-Art (SoA), this work addresses the problem of mapping dependent tasks on wireless nodes with DVFS to balance the energy consumption, under real-time and energy constraints. The main contributions of this paper are summarized as follows:

(1) Task triplication scheme and DVFS scheme are integrated into the task mapping process to enhance data transmission reliability and to exploit the trade-off between task execution time and energy consumption, respectively. We formulate the mapping of tasks on the wireless nodes as a Mixed-Integer Non-Linear Programming (MINLP), taking into account the application and the system constraints (e.g., task deadline, task dependencies, data routing, node type, and node energy). Nonlinear items are caused by the product of optimization variables related to frequency assignment, task allocation, and task triplication decisions. These items are replaced by auxiliary variables and additional linear constraints, so as to transform the MINLP problem into a Mixed-Integer Linear Programming (MILP), without degrading the quality of the solution.
(2) We propose an Optimal Reliability Task Mapping (ORTM) algorithm based on Benders decomposition to efficiently and optimally solve the transformed problem. The ORTM algorithm decomposes the original problem into two smaller easier-to-solve problems. It solves them iteratively, by using the solution of one problem in the other. The first problem is an Integer

Linear Programming (ILP) for task allocation, task triplication, and frequency assignment. The second problem is a Linear Programming (LP) for task scheduling. By iterating the problems, the optimality of the solution is guaranteed. We also design a novel Heuristic Reliability Task Mapping (HRTM) algorithm based on the greedy approach. Compared with the ORTM algorithm, the HRTM algorithm removes the iteration process and solves the problems sequentially. Therefore, the computation time of the HRTM algorithm is significantly reduced.

(3) Finally, we perform extensive simulations to analyze the solution quality, the computation time, and the scalability of the proposed task mapping scheme and the proposed ORTM and HRTM algorithms. The results show that the proposed task mapping scheme outperforms other task mapping schemes in terms of data transmission reliability and node energy efficiency. In addition, the ORTM algorithm finds the optimal solution with reduced computation time (24.84% on average), compared to SoA optimal methods, and the HRTM algorithm runs ∼100 times faster than the ORTM algorithm with an average cost of 26.35% in energy efficiency.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the system model, and Section 4 formulates the task mapping problem. Section 5 and Section 6 design the optimal and the heuristic algorithms, respectively. Finally, Section 7 shows the simulation results, and Section 8 concludes this work.

2 RELATED WORK

Table 1 depicts some representative SoA task mapping approaches targeting a single platform, i.e., an Embedded System (ES), and several distributed platforms, i.e., a Networked System (NS). The tasks can be Independent (I) or Dependent (D). The optimization variables include Task Allocation (TA), Task Scheduling (TS), Task Replicas (TR), and Frequency Assignment (FA). During the task mapping process, constraints regarding Energy Supply (E) and Real-Time (T) can exist. To enhance data reliability, methods based on task Replication (R) and State estimation (S) are used. According to different problem structures and system requirements, Optimal (O) and Heuristic (H) algorithms are designed to solve the corresponding problems.

For embedded systems, approaches exist that focus on minimizing the energy consumption or minimizing the task makespan under energy supply and real-time constraints [8, 13, 21, 22, 38]. Since the optimization variables of task allocation and frequency assignment are binary, the corresponding task mapping problems are usually described as Mixed-Integer Programming (MIP). For independent tasks, the task mapping problem is modeled as an MINLP in [22]. By relaxing the binary variables to continuous variables, the problem is transformed into a convex problem and is solved by the interior point method. A similar task mapping problem without DVFS but with task migration is studied in [38], where the MILP-based task mapping problem is first relaxed to two subproblems, and then the subproblems are solved by the polynomial-time methods. For dependent tasks, additional variables and constraints are introduced into the problem to describe the dependency between the tasks. The task mapping problem is described by an MINLP in [21]. The nonlinear items, which are caused by the quadratic function, can be approximated by a linear function, and then the relaxed MILP problem is solved by the B&B method [5]. DVFS is combined with Dynamic Power Management (DPM) in [8] to enhance energy efficiency. The task mapping problem is formulated as an MILP and is solved according to the commercial solver, such as CPLEX. In [13], each processor has a fixed frequency level, and, thus, the task mapping problem is an MILP, which can be optimally solved by a hybrid algorithm based on Benders decomposition [3]. However, reliability is not considered in these works.

Task replication techniques have been widely used in embedded systems to deal with task reliability problems [17, 42, 46]. By replicating the tasks, the system's reliability can be improved as

Table 1. Classification of some task mapping approaches

Ref.	*Task*		*Platform*		*Variables*				*Constraints*		*Reliability*		*Solution*	
	I	D	ES	NS	TA	TS	TR	FA	E	T	R	S	O	H
[8]		√	√		√	√		√	√	√			√	
[13]		√	√		√	√				√			√	
[21]		√	√		√	√		√	√	√			√	
[22]	√		√		√			√	√	√				√
[38]	√		√		√				√	√				√
[17]	√		√		√		√		√		√			√
[42]		√	√		√		√	√	√		√			√
[46]		√	√		√		√	√	√	√	√			√
[35]		√		√	√				√					√
[4]		√		√	√				√					√
[12]		√		√	√	√			√	√			√	
[24]		√		√	√				√					√
[41]		√		√	√	√		√	√	√				√
[28]		√		√								√	√	
[31]		√		√								√	√	
[44]		√		√								√	√	
[9]		√		√								√	√	
Prop.		√		√	√	√	√	√	√	√	√		√	√

it is extremely unlikely to have errors on two or more copies at the same time. In [17], full replication is used, i.e., each task is replicated once at least. With more tasks being replicated, higher reliability is achieved. To enhance task reliability and reduce energy consumption, energy-efficient fault-tolerant scheduling is designed in [42] considering parallel task execution on processors. A similar problem is considered in [46] and a bi-objective genetic algorithm is proposed to balance two conflicting objectives: low energy consumption and high task reliability. However, task mapping approaches, focusing on a single platform (embedded systems), usually assume that the data communication cost is very small compared to the task execution cost. Thus, no data communication cost is considered during task replication and mapping. However, the approaches focusing on networked systems, where the platforms are distributed, additional energy and time are required by the nodes for data transmission and reception, which restricts DVFS and task replication options.

For the network systems, since the tasks are usually dependent [4, 12, 24, 35, 41], it is crucial to reduce the communication and computation costs (energy and time) on the nodes. The energy-aware task mapping problem is considered in [4, 35] to minimize the energy consumption of the nodes, and the problem is described by an Integer Non-Linear Programming (INLP). The original problem is first transformed into an ILP, and then solved by the greedy algorithm. The problem of maximizing the overall network lifetime under energy supply or task deadline constraints is considered in [12, 24]. Based on the problem structure, a game-theoretic approach is proposed in [12] to perform distributed computing, and a heuristic is designed in [24] to enhance system response time. However, DVFS is not taken into account in the above studies. Some works consider nodes with DVFS capabilities. In [41], the node energy consumption is optimized by determining the frequency-to-task assignment and the task-to-node allocation. This problem is formulated as an MINLP and solved by heuristics. The above studies mainly focus on minimizing the energy consumption of the nodes, without considering reliability. The task results are assumed to be corrected if the execution and transmission of the tasks are finished before the deadlines.

Approaches exist that focus on detecting FDI attacks on networked systems, based on state estimation. In [28], a Kalman Filter (KF) is used to estimate system states and a χ^2-detector is employed to evaluate the discrepancies between the estimated data and the measured data, so as to detect the existence of FDI. In [31], based on the KF and χ^2-detector combination, a Linear-Quadratic-Gaussian (LQG) controller is used to improve the resilience of the system to against FDI. To improve the success rate of detecting the FDI attacks, a summation (SUM) detector is proposed in [44]. Compared with the χ^2 detector, the SUM detector not only utilizes the current compromise information but also collects all historical information to reveal the threat. The Probability Density Functions (PDF) of measurements and control commands, and their confidence intervals, are estimated using historical records in [9]. If the values exceed a certain level, the measurements/control commands are re-generated by a Maximum Likelihood Estimation (MLE). However, these methods are based on state estimation, requiring knowing the system model in advance (e.g., state transition matrix $\boldsymbol{A}$, input matrix $\boldsymbol{B}$, and measurement matrix $\boldsymbol{C}$), and thus, the accuracy of the used system model affects the stability of the system. Furthermore, real-time and energy supply constraints are not considered, and task allocation and scheduling decisions are fixed, since a remote Base Station (BS) is employed for processing the data from the wireless nodes, with the aim to minimize the estimation/control errors.

The aforementioned works address task mapping either on single platforms, focusing on minimizing energy consumption for computation, potentially including task replication, or on distributed platforms, minimizing the computation and the communication costs of the nodes for the networked systems. This work belongs to the second category and enhances the existing approaches by proposing a task mapping method that considers task replication and DVFS, under real-time and energy constraints, without requiring knowledge of the system model.

3 APPLICATIONS AND SYSTEM MODEL

3.1 Motivational Example

We will use the example of Fig. 1 to describe and motivate our approach. Fig. 1 depicts a part of a networked control system with six wireless nodes $\{\theta_1, \ldots, \theta_6\}$, used for applications such as smart grid [23], fire detection [34], home automation [45], and precision agriculture [2]. Nodes θ_1, θ_2 and θ_3 are equipped with sensors, while node θ_4 is equipped with an actuator to monitor and control the physical variables, e.g., the voltage, current or phase in the grid [28], the temperature of POIs (points of interest) of the forest region [34], the illumination intensity for the indoor environment [45], and the soil moisture in the agriculture [2]. The application needs to monitor the physical variables periodically, process the readings of the sensors to compute the required action, and adjust the outputs of the actuators to control the physical variables. In Fig. 1, τ_1 corresponds to the sensing task, τ_4 corresponds to the data processing task, and τ_7 is the control task. τ_4 requires the sensing data from τ_1 to compute the action of the actuator. τ_7 requires the output of τ_4 to adjust the actuator's output. Therefore, allocating tasks τ_1 and τ_7 is restricted to the nodes equipped with sensors and actuators, respectively. However, the data processing task τ_4 can be placed on any node. Note that, due to the network topology and the task dependencies, the task allocation decisions influence not only the communication and computation energy of the nodes but also the communication time between the tasks.

The FDI attack can be implemented by compromising the sensor communication network [33]. Therefore, when dependent tasks, e.g., τ_1 and τ_4, are executed on different nodes, e.g., θ_1 and θ_5, there is a possibility to attack the node and alternate τ_1's output, during its transmission to τ_4. This work combines the benefits of *task mapping* with *task triplication* to improve the reliability of the networked system against FDI attacks. Using task triplication, the data reliability is enhanced, and

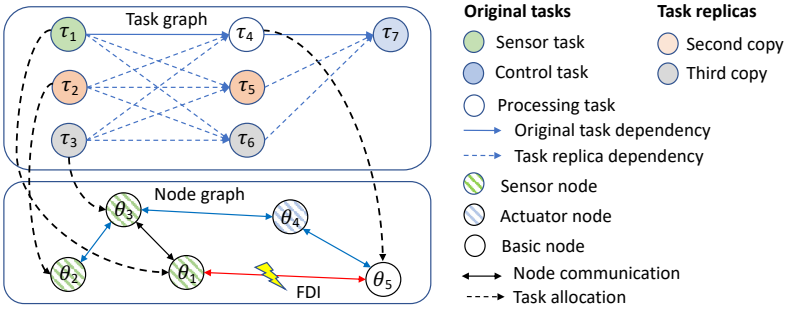


Fig. 1. Motivational example.

the influence caused by fault data can be mitigated [19], while knowledge of the system model is not required. The redundant information on the receiver node can be used to correct the information obtained by a compromised communication link. For instance, in Fig. 1, τ_2 and τ_3 are the second and the third replicas of τ_1, respectively. Under the task allocation $\{\tau_1 \rightarrow \theta_1, \tau_2 \rightarrow \theta_2, \tau_3 \rightarrow \theta_3, \tau_4 \rightarrow \theta_5\}$, two routing paths exist $\{\theta_2 \rightarrow \theta_3 \rightarrow \theta_4 \rightarrow \theta_5$ (blue), $\theta_1 \rightarrow \theta_5$ (red)$\}$. Therefore, θ_5 will receive task data (τ_1, τ_2 and τ_3) from θ_1 and θ_4. If θ_1 suffers from an FDI attack when it transmits the data of τ_1 to τ_4, the data can be corrected by comparing with the results of τ_2 and τ_3, before used as input to task τ_4. Executing task replicas on different nodes allows sending the information through different routing paths, dealing with FDI attacks with long duration. For instance, compared with the task allocation $\{\tau_2 \rightarrow \theta_3, \tau_3 \rightarrow \theta_3\}$, the task allocation $\{\tau_2 \rightarrow \theta_2, \tau_3 \rightarrow \theta_3\}$ has a longer time interval when θ_3 transmitting the data of τ_2 and τ_3, since a node will receive and transmit the task data in sequence.

Similarly, task τ_4 should be triplicated to enhance the reliability of the data that have to be transmitted to task τ_7. During the task mapping process, if dependent tasks τ_4 and τ_7 are executed on the same node (e.g., θ_4), there is no need to triplicate task τ_4, since no wireless transmission takes place between τ_4 and τ_7. However, not all dependent task can be assigned to the same node, due to the task requirements in specific sensors and actuators, e.g., $\{\tau_1 \rightarrow \theta_1, \tau_2 \rightarrow \theta_2, \tau_3 \rightarrow \theta_3, \tau_7 \rightarrow \theta_4\}$, the energy supply of the nodes and the real-time response of the application. Based on the above discussion, the task allocation and the task triplication problems are coupled and affected by real-time and energy constraints.

3.2 System Model

3.2.1 Task Model. We consider a real-time frame-based application that consists of a task set $\mathcal{T}$ with N periodic real-time tasks $\{\tau_1, \ldots, \tau_N\}$, which includes the original task, its second and third replicas. The tasks are considered dependent and non-preemptive [8, 21]. Each task τ_i is described by a tuple $\{e_i, t_i^a, t_i^s, d_i, p_i\}$. e_i is the number of execution cycles of task τ_i, measured in Worst-Case Execution Cycles (WCEC). t_i^a, t_i^s, d_i and p_i are the arrival time, the start time, the deadline, and the period of task τ_i, respectively. Without loss of generality, task arrival times are equal to the time instance 0. We assume that the deadline and the period of task τ_i are equal to H, which is given by the frame of the application. The proposed approach can be adapted for task sets with different periods, usually consisting of independent tasks. This can be achieved by unrolling the tasks in the hyper-period (i.e., the least common multiple of all task periods $\{p_1, \ldots, p_N\}$ [8, 29]), and setting the task arrival time and the task deadline, accordingly. The dependency between the tasks is given by a binary matrix $\boldsymbol{o} = [o_{ij}]_{N \times N}$. If $o_{ij} = 1$, τ_i precedes τ_j and τ_j is the closest task of

τ_i, else, $o_{ij} = 0$. Let s_{ij} denote the size of data that task τ_i generates for task τ_j. If tasks τ_i and τ_j are independent, we have $s_{ij} = 0$.

3.2.2 Node Model. We consider a networked system with M wireless nodes $\{\theta_1, \ldots, \theta_M\}$. The nodes have different capabilities, with $\{\theta_1, \ldots, \theta_{M_s}\}$ describing the nodes equipped with sensors and $\{\theta_{M_s+1}, \ldots, \theta_{M_s+M_a}\}$ describing the nodes equipped with actuators, and $M_s+M_a \leq M$. The processors of the nodes, where the computation takes place, use the same Instruction Set Architecture (ISA), and they support DVFS. Each processor has L different discrete Voltage/Frequency (V/F) levels $\{(v_1, f_1), \ldots, (v_L, f_L)\}$. As the relationship between voltage and frequency is almost linear [8, 10], when the supply voltage is changed, the clock frequency is changed accordingly. The execution time of task τ_i, when it is executed at frequency f_l on a processor, is given by e_i/f_l. The power consumption of a processor is modeled as the sum of static power P^s and dynamic power P^d, that is $P^c = P^s + P^d$ [8, 11]. The static power and dynamic power, under a given voltage/frequency level (v_l, f_l), are calculated as $P_l^s = C^s v_l^\rho$ and $P_l^d = C^d f_l v_l^2$, respectively, where C^s, ρ and C^d are the constants depending on the type of processor. We assume that a processor switches to the idle mode immediately when it has no task to execute, and the idle power is P^0. The time and energy required for the transition are incorporated into the task execution time and energy [22] since they are very small compared to the time and energy required for task execution.

4 PROBLEM FORMULATION

The aim of this work is to balance the energy consumption of the nodes, under real-time and energy constraints, exploiting task mapping, task triplication and DVFS. Therefore, we need to determine 1) which node the task should execute on (task allocation); 2) what voltage/frequency level should be used for the task (voltage/frequency assignment); 3) whether a task needs to be triplicated or not (task triplication); 4) when a task starts its execution (task scheduling). To formulate the task mapping problem, we introduce the following variables: 1) binary variable $q_{ik} = 1$ if task τ_i is allocated to node θ_k, else, $q_{ik} = 0$; 2) binary variable $c_{il} = 1$ if task τ_i is executed with the voltage/frequency (v_l, f_l), else, $c_{il} = 0$; 3) binary variable $h_i = 1$ if task τ_i is triplicated, else, $h_i = 0$; and 4) continuous variable t_i^s represents the start time of task τ_i.

We assume that the system runs with a given routing protocol. Therefore, the routing energy cost matrix $\boldsymbol{r} = [r_{mnk}]_{M\times M\times M}$ and the routing time matrix $\boldsymbol{t} = [t_{mn}]_{M\times M}$ are known in advance [35], where r_{mnk} is the energy consumed of node θ_k, when routing a unit of data from θ_m to θ_n, while t_{mn} is the time required to transmit unit of data from θ_m to θ_n. For presentation reasons, let $\mathcal{L} \triangleq \{1, \ldots, L\}$, $\mathcal{N} \triangleq \{1, \ldots, N\}$, $\mathcal{M} \triangleq \{1, \ldots, M\}$, $\mathcal{M}_s = \{1, \ldots, M_s\}$, $\mathcal{M}_a = \{M_s + 1, \ldots, M_s + M_a\}$, and $\mathcal{G} \triangleq \{1, \ldots, G\}$, where G is the number of original tasks. The main parameters and variables used in the problem formulation are summarized in Table 2.

4.1 Preliminary

4.1.1 Task Triplication Scheme. Since we consider task triplication and task triplication acts on original tasks, we initially analyze the relationship between the allocation and the triplication of original tasks. Let $\alpha_i = 3i - 2$ $(1 \leq i \leq G)$ denote the subscript of the original task. Therefore, tasks τ_{α_i+1} and τ_{α_i+2} are the second and third replicas (duplication and triplication) of task τ_{α_i}, respectively. If $h_{\alpha_i} = 0$, task τ_{α_i} is not triplicated, and thus, we have $e_{\alpha_i+1} = e_{\alpha_i+2} = 0$. As a result, tasks τ_{α_i+1} and τ_{α_i+2} do not exist. They can be removed from the task graph and will not generate any data for the successors or receive any data from the predecessors.

To determine the value of h_{α_i}, let's consider the simple example shown in Fig. 1, where τ_1, τ_4 and τ_7 are the original tasks. Let $\mathcal{S}_1 = \{\tau_1, \tau_4\}$ and $\mathcal{S}_2 = \{\tau_4, \tau_7\}$. Based on the elements in the task sets $\mathcal{S}_1$ and $\mathcal{S}_2$, we get the following four cases: 1) Tasks in $\mathcal{S}_1$ are allocated to different nodes, while

Table 2. Symbols used in the problem formulation

Parameters	
M_a	number of actuator nodes
M_s	number of sensor nodes
M	number of wireless nodes
G	number of original tasks
N	number of original tasks and replicas
L	number of voltage/frequency levels
θ_k	the kth node
τ_i	the ith task
(v_l, f_l)	the lth voltage/frequency level
H	scheduling horizon
P_l^s	static power of processor running on (v_l, f_l)
P_l^d	dynamic power of processor running on (v_l, f_l)
P^0	idle power of processor
E_k^r	energy supply of node θ_k
e_i	execution cycles of task τ_i
o_{ij}	$= \begin{cases} 1 & \text{if task } \tau_i \text{ proceeds } \tau_j \text{ and } \tau_j \text{ is the nearest task of } \tau_i \\ 0 & \text{else} \end{cases}$
N_{α_i}	number of tasks that dependent on task τ_{α_i}
s_{ij}	size of data that task τ_i produces for task τ_j
$r_{\beta\gamma k}$	energy consumed of node θ_k when routing unit of data from θ_β to θ_γ
$t_{\beta\gamma}$	time required to transmit unit of data from θ_β to θ_γ
Integer Variables	
q_{ik}	$= \begin{cases} 1 & \text{if task } \tau_i \text{ is allocated to node } \theta_k \\ 0 & \text{else} \end{cases}$
c_{il}	$= \begin{cases} 1 & \text{if task } \tau_i \text{ is executed with frequency } f_l \\ 0 & \text{else} \end{cases}$
u_{ij}	$= \begin{cases} 1 & \text{if task } \tau_i \text{ proceeds task } \tau_j \\ 0 & \text{else} \end{cases}$
h_i'	$= \begin{cases} 1 & \text{if task } \tau_i \text{ exists} \\ 0 & \text{else} \end{cases}$
Continuous Variables	
t_i^s	start time of task τ_i

tasks in $\mathcal{S}_2$ are allocated to the same node; 2) Tasks in $\mathcal{S}_1$ are allocated to different nodes and tasks in $\mathcal{S}_2$ are allocated to different nodes; 3) Tasks in $\mathcal{S}_1$ are allocated to the same node and tasks in $\mathcal{S}_2$ are allocated to the same node; 4) Tasks in $\mathcal{S}_1$ are allocated to the same node, while tasks in $\mathcal{S}_2$ are allocated to different nodes.

For $Case_1$, τ_1 is triplicated, while τ_4 is not triplicated ($h_1 = 1$ and $h_4 = 0$). For $Case_2$, τ_1 and τ_4 are triplicated ($h_1 = h_4 = 1$). For $Case_3$, τ_1 and τ_4 are not triplicated ($h_1 = h_4 = 0$). For $Case_4$, τ_4 requires triplication, since τ_4 and τ_7 are allocated to different nodes. In addition, τ_1 is triplicated, as τ_4, τ_5 and τ_6 are allocated to different nodes. The decision regarding the tasks in $\mathcal{S}_2$ propagates to τ_1, which needs to be also triplicated, even if the tasks in $\mathcal{S}_1$ are allocated to the same node. From the above

example, we observe that the triplication decision is not only related to the allocation of original tasks, but also to the allocation of the replicas.

To formulate the relation between task triplication and task allocation decisions, we introduce a parameter N_{α_i} and an auxiliary variable M_{α_i} for the original task τ_{α_i}, where N_{α_i} represents the number of tasks that dependent on task τ_{α_i}, while M_{α_i} represents the number of dependent tasks that are assigned to the same node as task τ_{α_i}. Based on the dependency between the tasks, the values of N_{α_i} and M_{α_i} are given by:

$$N_{\alpha_i} = \sum\nolimits_{j \in \mathcal{N}} o_{\alpha_i j}, \ \forall i \in \mathcal{G}, \tag{1}$$

$$M_{\alpha_i} = \sum\nolimits_{j \in \mathcal{N}} \sum\nolimits_{k \in \mathcal{M}} o_{\alpha_i j} q_{\alpha_i k} q_{jk}, \ \forall i \in \mathcal{G}, \tag{2}$$

In Fig. 1, as τ_1 and τ_4 are original tasks, we have $N_1 = 3$ and $N_4 = 1$ for both cases due to

$$\boldsymbol{o} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

For $Case_1$, we have $M_1 \leq 2$ as τ_1 and τ_4 are assigned to different nodes, and $M_4 = 1$ as τ_4 and τ_7 are assigned to the same node. Similarly, we get $M_1 \leq 2$ and $M_4 = 0$ for $Case_2$; $M_1 = 3$ and $M_4 = 1$ for $Case_3$; $M_1 \leq 2$ and $M_4 = 0$ for $Case_4$. Since $M_{\alpha_i} \leq N_{\alpha_i}$, based on the triplication decisions regarding the tasks τ_1 and τ_4, we can get the following conclusion: if $M_{\alpha_i} = N_{\alpha_i}$, $h_{\alpha_i} = 0$, else (i.e., $M_{\alpha_i} \leq N_{\alpha_i} - 1$), $h_{\alpha_i} = 1$. Note that this $if - else$ description is not a standard constraint and M_{α_i} is changed with task allocation variable q_{ik}. We introduce the following lemma to determine the value of h_{α_i}.

Lemma 4.1. *Let x and y denote an integer variable and a binary variable, respectively, where x is bounded by $0 \leq x \leq m$. $m \geq 0$ and $n \geq 0$ are the positive integer constants. The case: if $x = n \rightarrow y = 0$, else (i.e., $x \leq n - 1$), $y = 1$ can be given by the linear constraint $\frac{x-n+1}{m+1} \leq 1 - y \leq \frac{x+1}{n+1}$.*

Proof. If $x = n$, we have $\frac{x-n+1}{m+1} > 0$ and $\frac{x+1}{n+1} = 1$. Therefore, we get $y = 0$. On the other hand, if $x \leq n - 1$, since $\frac{x-n+1}{m+1} \leq 0$ and $\frac{x+1}{n+1} < 1$, we obtain $y = 1$. □

According to *Lemma* 4.1, the task triplication decision h_{α_i} is determined by the following constraint:

$$\frac{M_{\alpha_i} - N_{\alpha_i} + 1}{N + 1} \leq 1 - h_{\alpha_i} \leq \frac{M_{\alpha_i} + 1}{N_{\alpha_i} + 1}, \ \forall i \in \mathcal{G}, \tag{3}$$

where N_{α_i} and M_{α_i} are given by Eq. (1) and Eq. (2), respectively.

4.1.2 Node Communication Cost. Let $\boldsymbol{h}' = [h'_1, \ldots, h'_{3G}]^T = [1, h_1, h_1, \ldots, 1, h_{\alpha_i}, h_{\alpha_i}, \ldots, 1, h_G, h_G]^T$. With the variable h'_i, the number of execution cycles of task τ_i is $h'_i e_i$ and the size of data, that task τ_i generate for task τ_j, is $h'_i h'_j s_{ij}$. If task τ_i is assigned to node θ_m, while task τ_j $(i \neq j)$ is assigned to node θ_n $(m \neq n)$, the communication energy consumed by node θ_k $(k \neq m \neq n)$ to send the data from τ_i to τ_j is $h'_i h'_j s_{ij} q_{im} q_{jn} r_{mnk}$. Therefore, the communication energy of node θ_k is

$$E_k^t = \sum\nolimits_{i \in \mathcal{N}} \sum\nolimits_{j \in \mathcal{N}} \sum\nolimits_{m \in \mathcal{M}} \sum\nolimits_{n \in \mathcal{M}} h'_i h'_j s_{ij} q_{im} q_{jn} r_{mnk}, \ \forall k \in \mathcal{M}. \tag{4}$$

To execute a task τ_j, we must collect all the data generated from its previous dependent tasks. If task τ_i is assigned to node θ_m, while task τ_j is assigned to node θ_n, the time required to transmit the

data with size $h'_i h'_j s_{ij}$, from θ_m to θ_n, is $h'_i h'_j s_{ij} q_{im} q_{jn} t_{mn}$. We assume that the used communication protocol prevents concurrent reception of messages from multiple nodes, avoiding data conflicts. Hence, the time required to collect the data for the execution of task τ_j is

$$t_j^r = \sum_{i\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{M}} h'_i h'_j s_{ij} q_{im} q_{jn} t_{mn}, \ \forall j \in \mathcal{N}. \tag{5}$$

4.1.3 Task Execution Cost. Since the voltage/frequency level is adjusted in a discrete manner, similar to the previously published work [22], the time and the energy required to execute task τ_i with $h'_i e_i$ cycles are $t_i^c = \sum_{l\in\mathcal{L}} c_{il} \frac{h'_i e_i}{f_l}$ and $E_i^c = \sum_{l\in\mathcal{L}} c_{il} \frac{h'_i e_i}{f_l} P_l^c$, respectively. On this basis, considering the task allocation decision q_{ik}, the time and the energy required to execute the tasks assigned to node θ_k are

$$t_k^n = \sum_{i\in\mathcal{N}} q_{ik} t_i^c = \sum_{i\in\mathcal{N}} \sum_{l\in\mathcal{L}} q_{ik} c_{il} \frac{h'_i e_i}{f_l}, \ \forall k \in \mathcal{M}, \tag{6}$$

$$E_k^n = \sum_{i\in\mathcal{N}} q_{ik} E_i^c = \sum_{i\in\mathcal{N}} \sum_{l\in\mathcal{L}} q_{ik} c_{il} \frac{h'_i e_i}{f_l} P_l^c, \ \forall k \in \mathcal{M}. \tag{7}$$

Therefore, the computation energy of node θ_k is

$$E_k^d = E_k^n + (H - t_k^n) P^0, \ \forall k \in \mathcal{M}, \tag{8}$$

where $H - t_k^n$ is the idle time of the processor during the scheduling horizon H.

4.2 Primal Problem

To balance the energy consumption of the nodes and increase the system lifetime and connectivity, we can minimize the maximum energy consumption on a single node. Thus, the objective function is set to $F = \max_{\forall k\in\mathcal{M}}\{(E_k^t + E_k^d)/E_k^{ini}\}$. With this aim, the task mapping problem (Primal Problem, PP) has the form

$$\mathbf{PP}: \min_{\boldsymbol{q},\boldsymbol{c},\boldsymbol{u},\boldsymbol{h}',\boldsymbol{t}^s} F \quad \text{s.t. } (3), (10) - (19). \tag{9}$$

The constraints of the problem are described as follows:

(1) *Task Allocation Constraints*: Note that each task τ_i is assigned to one node (no task migration), and the tasks requiring nodes equipped with the sensor or actuator are restricted. The original task τ_{α_i} and its replicas τ_{α_i+1} and τ_{α_i+2} should not be assigned to the same node. Therefore, we have

$$\sum_{k\in\mathcal{M}} q_{ik} = 1, \ \forall i \in \mathcal{N}, \tag{10}$$

$$q_{ik} = 1, \ \forall (i,k) \in \mathcal{D}, \tag{11}$$

$$0 \le q_{\alpha_i k} + q_{\alpha_i+1 k} + q_{\alpha_i+2 k} \le 1, \ \forall i \in \mathcal{G}, \ \forall k \in \mathcal{M}, \tag{12}$$

where $\mathcal{D}$ is the set of task and node indexes that have restricted task allocation.

(2) *Frequency Assignment Constraint*: Since each task τ_i is executed with one voltage/frequency level, we get

$$\sum_{l\in\mathcal{L}} c_{il} = 1, \ \forall i \in \mathcal{N}. \tag{13}$$

(3) *Task Sequence Constraint*: For the dependent tasks (e.g., tasks τ_1, τ_4 and τ_7 in Fig. 1), no matter if they are assigned to the same node or different nodes, the start time and the end time of tasks is bounded. Therefore, we have

$$t_j^s + (1 - o_{ij})H \ge t_i^s + o_{ij}(t_i^e - t_i^s) + t_j^r, \ \forall i \ne j \in \mathcal{N}, \tag{14}$$

where t_i^s and t_i^e are the start time and the end time of task τ_i, respectively ($0 \le t_i^s \le t_i^e \le H$). Therefore, the task execution time is given by $t_i^c = t_i^e - t_i^s$. If $o_{ij} = 1$ (i.e., τ_i precedes τ_j and τ_j is the closest task of τ_i), we have $t_j^s \ge t_i^e + t_j^r$, else (i.e., $o_{ij} = 0$), Eq. (14) is always satisfied.

(4) *Task Non-Preemption Constraint*: Note that some tasks can be independent in the task set (e.g., tasks τ_1, τ_2 and τ_3 in Fig. 1). When independent tasks are assigned to the same node, we need to determine their execution sequence. To this end, we introduce a binary variable u_{ij}. Since each node executes no more than one task at the same time, we get

$$t_i^e \le t_j^s + (2 - q_{ik} - q_{jk})H + (1 - u_{ij})H, \ \forall i \ne j \in \mathcal{N}, \ \forall k \in \mathcal{M}. \tag{15}$$

If tasks τ_i and τ_j are assigned to the same node (e.g., $q_{ik} = q_{jk} = 1$), Eq. (15) is meaningful, else, Eq. (15) is always true. With $q_{ik} = q_{jk} = 1$, if $u_{ij} = 1$ (i.e., τ_i precedes τ_j), we have $t_i^e \le t_j^s$, else (i.e., $u_{ij} = 0$), Eq. (15) is always satisfied.

(5) *Task Deadline Constraint*: The system is periodic, i.e., it operates in rounds [35] (scheduling horizon H), where in each round, all the tasks are executed once. Therefore, the constraint related to the task deadline is

$$t_i^e \le H, \ \forall i \in \mathcal{N}. \tag{16}$$

(6) *Energy Constraints*: Let E_k^s, E_k^a and E_k^r denote the sensing energy of the sensor, the acting energy of the actuator, and the energy supply of the node at the rth round, respectively. Since the total energy consumed by node θ_k, during the scheduling horizon H, should not exceed its energy supply, we get

$$E_k^t + E_k^d + E_k^s \le E_k^r, \ \forall k \in \mathcal{M}_s, \tag{17}$$

$$E_k^t + E_k^d + E_k^a \le E_k^r, \ \forall k \in \mathcal{M}_a, \tag{18}$$

$$E_k^t + E_k^d \le E_k^r, \ \forall k \in \mathcal{M}, \ \forall k \notin \mathcal{M}_s, \mathcal{M}_a. \tag{19}$$

4.3 Problem Linearization

Due to the products of binary variables (e.g., $q_{im}q_{jn}$ in Eq. (3), $h_i'h_j'q_{im}q_{jn}$ and $q_{ik}c_{il}h_i'$ in Eq. (14), Eq. (16)–Eq. (19)), the PP is an MINLP problem, which makes it challenging to solve.

Theorem 4.1. *The task mapping problem of the networked system (i.e., PP) is $\mathcal{NP}$-hard.*

Proof. Please refer to [7] for the details. □

Linearization is an efficient way to simplify the structure of the problem. To linearize the nonlinear items, we introduce the following lemma.

Lemma 4.2. *Let x, y, and z denote the binary variables. The nonlinear item $z = xy$ can be linearized as follows: $z \le x$, $z \le y$ and $z \ge x + y - 1$.*

Proof. The inequalities $z \le x$ and $z \le y$ ensure that $z = 0$ if either $x = 0$ or $y = 0$. The inequality $z \ge x + y - 1$ makes sure that $z = 1$ if both x and y are set to 1. □

According to *Lemma 4.2*, we first introduce an auxiliary (binary) variable β_{imjn} to replace the nonlinear term $q_{im}q_{jn}$, and then add the following constraints into the PP:

$$\{\beta_{imjn} \le q_{im}, \ \beta_{imjn} \le q_{jn}, \ \beta_{imjn} \ge q_{im} + q_{jn} - 1\}, \ \forall i \ne j \in \mathcal{N}, \ \forall m, n \in \mathcal{M}. \tag{20}$$

Therefore, we have

$$M_{\alpha_i} = \sum\nolimits_{j \in \mathcal{N}} \sum\nolimits_{k \in \mathcal{M}} o_{\alpha_i j} \beta_{\alpha_i k j}. \tag{21}$$

On this basis, to deal with the nonlinear item $h'_i h'_j q_{im} q_{jn}$, we reuse the auxiliary variable β_{imjn}. Let $\gamma_{ij} = h'_i h'_j$ and $\delta_{imjn} = \gamma_{ij}\beta_{imjn}$. Therefore, if the following constraints are added to the PP:

$$\{\gamma_{ij} \le h'_i,\ \gamma_{ij} \le h'_j,\ \gamma_{ij} \ge h'_i + h'_j - 1\},\ \forall i \ne j \in \mathcal{N}, \tag{22}$$

$$\{\delta_{imjn} \le \gamma_{ij},\ \delta_{imjn} \le \beta_{imjn},\ \delta_{imjn} \ge \gamma_{ij} + \beta_{imjn} - 1\},\ \forall i \ne j \in \mathcal{N},\ \forall m, n \in \mathcal{M}, \tag{23}$$

Eq. (4) and Eq. (5) can be rewritten as follows:

$$E_k^t = \sum\nolimits_{i\in\mathcal{N}} \sum\nolimits_{j\in\mathcal{N}} \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{n\in\mathcal{M}} s_{ij} r_{mnk} \delta_{imjn},\ \forall k \in \mathcal{M}, \tag{24}$$

$$t_j^r = \sum\nolimits_{i\in\mathcal{N}} \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{n\in\mathcal{M}} s_{ij} t_{mn} \delta_{imjn},\ \forall j \in \mathcal{N}. \tag{25}$$

Similarly, let $\varepsilon_{il} = h'_i c_{il}$ and $\eta_{ilk} = \varepsilon_{il} q_{ik}$. The nonlinear items $h'_i c_{il}$ and $h'_i c_{il} q_{ik}$ can be replaced by the auxiliary (binary) variables ε_{il} and η_{ilk} and the following constraints:

$$\{\varepsilon_{il} \le h'_i,\ \varepsilon_{il} \le c_{il},\ \varepsilon_{il} \ge h'_i + c_{il} - 1\},\ \forall i \in \mathcal{N},\ \forall l \in \mathcal{L}, \tag{26}$$

$$\{\eta_{ilk} \le \varepsilon_{il},\ \eta_{ilk} \le q_{ik},\ \eta_{ilk} \ge \varepsilon_{il} + q_{ik} - 1\},\ \forall i \in \mathcal{N},\ \forall l \in \mathcal{L},\ \forall k \in \mathcal{M}. \tag{27}$$

Therefore, Eq. (6) and Eq. (7) can be rewritten as follows:

$$t_k^n = \sum\nolimits_{i\in\mathcal{N}} \sum\nolimits_{l\in\mathcal{L}} \eta_{ilk} \frac{e_i}{f_l},\ \forall k \in \mathcal{M}, \tag{28}$$

$$E_k^n = \sum\nolimits_{i\in\mathcal{N}} \sum\nolimits_{l\in\mathcal{L}} \eta_{ilk} \frac{e_i}{f_l} P_l^c,\ \forall k \in \mathcal{M}. \tag{29}$$

Based on the above analysis, the PP is linearized as follows:

$$\mathbf{PP1}: \min_{\substack{\boldsymbol{q},\boldsymbol{c},\boldsymbol{u},\boldsymbol{h}',\boldsymbol{t}^s,\\ \boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta},\boldsymbol{\varepsilon},\boldsymbol{\eta},\xi}} \xi \tag{30}$$

$$\text{s.t.} \begin{cases} \frac{E_k^t + E_k^d}{E_k^{ini}} \le \xi,\ \forall k \in \mathcal{M}, \\ (3), (10) - (19), (20), (22), (23), (26), (27), \end{cases}$$

where ξ is an auxiliary variable, and M_{α_i}, E_k^t, t_j^r, t_k^n, and E_k^n in constraints are given by Eqs. (21), (24), (25), (28) and (29), respectively.

5 OPTIMAL TASK DEPLOYMENT ALGORITHM

In this section, we propose an Optimal Reliability Task Mapping algorithm, referred to as ORTM, to find the optimal solution for PP1. The basic idea of the ORTM algorithm comes from the Benders decomposition [3]. Instead of considering all the variables and constraints simultaneously, the ORTM algorithm separates the PP1 into smaller, easier-to-solve problems, the Master Problem (MP) and the Slave Problem (SP). Then, the PP1 is solved by iteratively solving the MP and the SP. Note that the PP1 has the following abstract form

$$\mathbf{PP2}: \min_{\boldsymbol{x},\boldsymbol{y}} \Phi = \boldsymbol{f}^T \boldsymbol{x} \tag{31}$$

$$\text{s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1, \\ \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2, \end{cases}$$

where $\boldsymbol{x} \triangleq (\boldsymbol{q}, \boldsymbol{c}, \boldsymbol{u}, \boldsymbol{h}', \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\varepsilon}, \boldsymbol{\eta})$ and $\boldsymbol{y} \triangleq (\boldsymbol{t}^s, \xi)$ denote the vectors of binary and continuous variables, respectively. The vector $\boldsymbol{f}$ represents the coefficients in the objective function. The matrices $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are the coefficients in the constraints. $\boldsymbol{b}_1$ ($\boldsymbol{b}_2$) is a u (v)-dimensional vector.

Based on the structure of PP2, at the kth iteration, the MP and the SP are formulated as follows:

$$\mathbf{MP}: \Phi_l(k) = \min_{\boldsymbol{x},\hat{\Phi}} \ \hat{\Phi} \tag{32}$$
$$\text{s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1, \\ \boldsymbol{C}_1 : \hat{\Phi} \geq (\boldsymbol{f}^T + \boldsymbol{\lambda}^T(i)\boldsymbol{C})\boldsymbol{x} - \boldsymbol{\lambda}^T(i)\boldsymbol{b}_2, \ 1 \leq i \leq m, \\ \boldsymbol{C}_2 : 0 \geq \hat{\boldsymbol{\lambda}}^T(j)(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2), \ 1 \leq j \leq n, \end{cases}$$

$$\mathbf{SP}: \Phi_u(k) = \min_{\boldsymbol{y}\geq 0} \ \boldsymbol{f}^T\boldsymbol{x}(k) \tag{33}$$
$$\text{s.t. } \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2,$$

where $\hat{\Phi}$ is an auxiliary variable to facilitate the iterations between MP and SP, and $\boldsymbol{x}(k)$ is the MP solution at the kth iteration. $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are the sets of constraints called Benders cuts ($m + n = k$). By solving the MP and the SP, we obtain a lower bound, $\Phi_l(k)$, and an upper bound, $\Phi_u(k)$, of Φ^* [36], where Φ^* is the optimal value of Φ. The iteration stops when $\Phi_u(k) - \Phi_l(k) \leq \varepsilon$.

The MP accounts for all the binary variables $\boldsymbol{x}$ and the associated portion of the objective function and the constraints of the PP2. The SP includes all the continuous variables $\boldsymbol{y}$ and the associated constraints in the PP2. By solving the Dual Slave Problem (DSP)

$$\mathbf{DSP}: \max_{\boldsymbol{\lambda}\geq 0} \ (\boldsymbol{f}^T + \boldsymbol{\lambda}^T\boldsymbol{C})\boldsymbol{x}(k) - \boldsymbol{\lambda}^T\boldsymbol{b}_2 \tag{34}$$
$$\text{s.t. } \boldsymbol{D}'\boldsymbol{\lambda} \geq 0,$$

we obtain the information about the SP portion of the PP2, and this information is communicated to the MP via Benders cuts, where $\boldsymbol{\lambda} \triangleq [\lambda_i]$ are the Lagrange multipliers.

Based on the solution of the DSP, we have the following three cases.

(1) If DSP is infeasible, PP2 has no feasible solution.
(2) If DSP has a bounded solution $\boldsymbol{\lambda}(k)$, since 1) $\hat{\Phi}(k) < (\boldsymbol{f}^T + \boldsymbol{\lambda}^T(k)\boldsymbol{C})\boldsymbol{x}(k) - \boldsymbol{\lambda}^T(k)\boldsymbol{b}_2$, where $\hat{\Phi}(k)$ is the optimal solution of MP at the kth iteration, and 2) $\boldsymbol{x}(k)$ is not the optimal solution of PP2 due to $\Phi_u(k) - \Phi_l(k) > \varepsilon$, the non-optimal solution $\boldsymbol{x}(k)$ is excluded by the constraint $\hat{\Phi} \geq (\boldsymbol{f}^T + \boldsymbol{\lambda}^T(k)\boldsymbol{C})\boldsymbol{x} - \boldsymbol{\lambda}^T(k)\boldsymbol{b}_2$.
(3) If DSP has an unbounded solution, the SP has no feasible solution under the given solution $\boldsymbol{x}(k)$. However, the SP is feasible if positive variables $\boldsymbol{\xi} \triangleq [\xi_i]$ are introduced to relax the constraints. Based on this idea, we construct a Feasibility Check Problem (FCP) and solve its dual problem (DFCP).

$$\mathbf{FCP}: \min_{\boldsymbol{y},\boldsymbol{\xi}\geq 0} \ \mathbf{1}^T\boldsymbol{\xi} \tag{35}$$
$$\text{s.t. } \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2 + \boldsymbol{\xi}.$$

$$\mathbf{DFCP}: \max_{\hat{\boldsymbol{\lambda}}\geq 0} \ \hat{\boldsymbol{\lambda}}^T(\boldsymbol{C}\boldsymbol{x}(k) - \boldsymbol{b}_2) \tag{36}$$
$$\text{s.t.} \begin{cases} \mathbf{1} - \hat{\boldsymbol{\lambda}} \geq 0, \\ \boldsymbol{D}^T\hat{\boldsymbol{\lambda}} \geq 0, \end{cases}$$

where $\hat{\boldsymbol{\lambda}} \triangleq [\hat{\lambda}_i]$ are the Lagrange multipliers. Let $\boldsymbol{\xi}(k)$ and $\hat{\boldsymbol{\lambda}}(k)$ be the optimal solutions of FCP and DFCP at the kth iteration, respectively. If the SP is infeasible, this implies that some constraints cannot be satisfied, and thus, their related relaxation variables are non-zero. We have $\mathbf{1}^T\boldsymbol{\xi}(k) > 0$, and further, $\mathbf{1}^T\boldsymbol{\xi}(k) = \hat{\boldsymbol{\lambda}}^T(k)(\boldsymbol{C}\boldsymbol{x}(k) - \boldsymbol{b}_2) > 0$ due to the strong duality [6]. Therefore, the infeasible solution $\boldsymbol{x}(k)$ can be excluded by the constraint $0 \geq \hat{\boldsymbol{\lambda}}^T(k)(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$.

With the constraint $\hat{\Phi} \geq (\boldsymbol{f}^T + \boldsymbol{\lambda}^T(k)\boldsymbol{C})\boldsymbol{x} - \boldsymbol{\lambda}^T(k)\boldsymbol{b}_2$ added into $\boldsymbol{C}_1$, or the constraint $0 \geq \hat{\boldsymbol{\lambda}}^T(k)(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ added into $\boldsymbol{C}_2$, at the $(k+1)^{th}$ iteration, the MP is solved again to obtain a new solution $\boldsymbol{x}(k+1)$ for the next iteration.

Theorem 5.1. *With a new constraint added to the MP at each iteration, the solution obtained by ORTM converges to the global optimal value Φ^* within a finite number of iterations.*

Proof. Since the non-optimal and the infeasible solutions of the binary variables $\boldsymbol{x}$ are excluded by the constraints $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$, and the dimension of $\boldsymbol{x}$ is finite, the solution converges to the global optimal value within a finite number of iterations. □

6 HEURISTIC APPROACH

Note that the ORTM algorithm generates a new constraint at each iteration and adds it to the ILP-based MP. With the number of iterations increasing, the computational complexity and the size of MP also increase. Therefore, the computational complexity of the ORTM algorithm is determined by the cost of solving the MP. Although the solution provided by the ORTM algorithm is optimal, this method cannot solve problems with large network sizes efficiently. To enhance the scalability of the proposed method, we propose a novel Heuristic Reliability Task Mapping algorithm, referred to as HRTM, to efficiently solve the PP. The reasons why we design the heuristic approach for PP, rather than PP1, are the following: 1) PP and PP1 are equivalent, 2) PP contains fewer variables, which makes the problem easier to solve, and 3) the heuristic approach determines the values of the variables sequentially, and thus, some nonlinear items, which are caused by the products of variables, can be omitted.

Similar to the ORTM algorithm, the HRTM algorithm is also based on problem decomposition. However, the main difference is that the subproblems in the ORTM algorithm are solved iteratively, while the subproblems in the HRTM algorithm are solved sequentially without iteration. Therefore, the computation time of the HRTM algorithm can be largely reduced. Based on the structure of PP, the HRTM algorithm contains two steps. We first consider the frequency assignment problem, in order to balance the task execution energy. Based on the solution of the first step, we solve the task mapping problem to balance the nodes' computation and communication energy.

6.1 Frequency Assignment Problem

The processors of the nodes have the same voltage/frequency levels $\{(v_1, f_1), \ldots, (v_F, f_L)\}$. Therefore, the frequency-to-task assignment is independent of the task-to-node allocation, and thus, it can be considered as the first step; the solutions in the next heuristic step will not violate the solution of this step. Regarding task triplication, the task allocation decision q_{ik} is unknown at this step, and according to Eq. (3), the task triplication decision h'_i is determined by the value of q_{ik}. Hence, we consider the worst case, where each original task τ_{α_i} is triplicated ($h_{\alpha_i} = 1$), i.e., the number of tasks is N.

Under the given frequency assignment decision c_{il}, the time and the energy required to execute task τ_i with e_i cycles are $t_i^c = \sum_{l \in \mathcal{L}} c_{il} \frac{e_i}{f_l}$ and $E_i^c = \sum_{l \in \mathcal{L}} c_{il} \frac{e_i}{f_l} P_l^c$, respectively. If a lower voltage/frequency level is used to execute the tasks, the task execution energy will be reduced. However, it will have an impact on the real-time execution, since the task execution time increases. Therefore, at the current step, we aim to balance the task execution energy, while guaranteeing that the real-time task constraint is satisfied. On the one hand, minimizing the maximum energy consumption on a single task's execution (i.e., $\min(\max_{\forall i \in \mathcal{N}} E_i^c)$) facilitates the balance of node energy consumption through task allocation, which is the main purpose of next step. On the other hand, taking the task real-time constraint into account, we can choose the proper voltage/frequency to execute the tasks to avoid missing the task deadlines.

Algorithm 1: Frequency-to-task assignment

```
1: Initialize: S[i] = −1, ∀i ∈ N;
2: Sort tasks {τ_1, ..., τ_N} according to AssSequence;
3: for ∀i ∈ N do
4:     MinMaxCompE = ∞;
5:     flag = −1;
6:     for ∀l ∈ L do
7:         S[i] = l;
8:         Calculate t_i^e based on c_il and o_ij;
9:         MaxComp = MaxCompE;
10:        if t_i^e > H then
11:            c_il = 0;
12:            Continue;
13:        else
14:            if MaxComp < MinMaxCompE then
15:                MinMaxCompE = MaxComp;
16:                flag = l;
17:            else
18:                MinMaxCompE = MinMaxCompE;
19:                flag = flag;
20:            end if
21:        end if
22:    end for
23:    if flag == −1 then
24:        Fail to assign frequency for task τ_i;
25:    else
26:        S[i] = flag;
27:    end if
28: end for
29: Calculate c_il according to S[i];
```

Based on the objective function and the constraints mentioned above, the Frequency Assignment Problem (FAP) is formulated as

$$\textbf{FAP}: \min_{\boldsymbol{c}} \left(\max_{\forall i \in \mathcal{N}} E_i^c \right) \quad \text{s.t. (13), (16).} \tag{37}$$

By using the idea of Greedy Algorithm (GA) [26, 35], we propose Algorithms 1 and 2 to solve this problem. For presentation reasons, a frequency assignment symbol $S[i]$ is introduced for each task τ_i, where $S[i] = l$ represents frequency f_l is used to execute task τ_i.

Before executing Algorithm 1, we need to determine the frequency assignment sequence of the tasks, *AssSequence*, required for sorting the tasks (Line 2). Note that the task start time and the node communication costs (energy and time) are unknown. To take real-time constraint (16) into account, we divide the tasks into several layers; tasks in different layers are dependent, while the tasks in the same layer are independent. Each task can identify its parent tasks based on the dependency between the tasks. On this basis, we can determine the layer to which a task should belong. The sequence is obtained following the layer order, and when independent tasks exist within a layer, they are sorted in descending order of execution cycles.

Algorithm 2: Calculate *MaxCompE*

```
1: MaxCompE = 0;
2: for ∀i ∈ N (S[i] ≠ −1) do
3:     CompE = (e_i / f_{S[i]}) P^c_{S[i]};
4:     if CompE > MaxCompE then
5:         MaxCompE = CompE;
6:     else
7:         MaxCompE = MaxCompE;
8:     end if
9: end for
```

To illustrate the above procedure, Fig. 2 shows a simple case with five dependent tasks. The task dependency is described by a binary matrix $\boldsymbol{o} = [o_{ij}]_{5\times5}$, where $o_{12} = o_{13} = o_{14} = o_{23} = o_{24} = o_{35} = o_{45} = 1$. According to the matrix $\boldsymbol{o}$, we can describe each task τ_i with the sets $PTIndex_i$ and $PTLayer_i$, where $PTIndex_i$ is the set of parent tasks and $PTLayer_i$ is the set of the layers assigned to parent tasks. For example, task τ_3 has two parent tasks τ_1 and τ_2, and the layers of tasks τ_1 and τ_2 are L_1 and L_2, respectively. Hence, we have $PTIndex_3 = \{1, 2\}$ and $PTLayer_3 = \{1, 2\}$, and thus, task τ_3 is assigned to layer L_3. On the other hand, the independent tasks τ_3 and τ_4 can be assigned to the same layer L_3. For tasks τ_3 and τ_4, assuming that the execution cycles of τ_3 are higher than those of τ_4 (i.e., $e_3 > e_4$), the τ_3 is considered first. Therefore, we obtain a frequency assignment sequence: $AssSequence = \{\tau_1 \rightarrow \tau_2 \rightarrow \tau_3 \rightarrow \tau_4 \rightarrow \tau_5\}$. Since the original task τ_{α_i} and its replicas τ_{α_i+1} and τ_{α_i+2} are independent, they are assigned to the same layer. Note that τ_{α_i}, τ_{α_i+1} and τ_{α_i+2} have the same execution cycles (i.e., $e_{\alpha_i} = e_{\alpha_i+1} = e_{\alpha_i+2}$). We assume that the frequency assignment sequence between them is $\tau_{\alpha_i} \rightarrow \tau_{\alpha_i+1} \rightarrow \tau_{\alpha_i+2}$. On the other hand, for the original tasks τ_{α_i} and τ_{α_j} $(i \neq j)$, if they are in the same layer and they have the same execution cycles (i.e., $e_{\alpha_i} = e_{\alpha_j}$), we consider that their frequency assignment priorities can be randomly chosen.

Based on the frequency assignment sequence *AssSequence*, Algorithm 1 finds the proper frequency for task τ_i under the real-time constraint, i.e., the selected frequency should cause the minimum increase in task execution energy, among the tasks that have already been assigned their frequency. During this process, Algorithm 2 is used to calculate the maximum energy required to execute a task, i.e., *MaxCompE*. If the real-time constraint cannot be satisfied under the given assignment decision $S[i] = l$, this decision will be excluded (Lines 10–12). As the task allocation decision is unknown, we omit the communication time of the nodes and assume that there is no gap between two adjacent tasks when calculating the end time of each task (Line 8). As the example shown in Fig. 2, we have $t_1^s = 0$, $t_1^e = t_1^c$, $t_2^e = t_1^e + t_2^c$, $t_3^e = \max\{t_1^e, t_2^e\} + t_3^c$, $t_4^e = \max\{t_1^e, t_2^e\} + t_4^c$, and $t_5^e = \max\{t_3^e, t_4^e\} + t_5^c$, where t_i^s, $t_i^c = \sum_{l\in\mathcal{L}} c_{il}\frac{e_i}{f_l}$ and $t_i^e = t_i^s + t_i^c$ are the start time, execution time and end time of task τ_i, respectively. By applying the above frequency selection process for each task τ_i (Lines 3–28), the solution to the FAP is found.

6.2 Task Mapping Problem

From Eq. (4) and Eq. (7), we observe that when the frequency assignment decision c_{il} is fixed, the communication cost E_k^t and the computation cost E_k^n of each node θ_k are determined by the task allocation decision q_{ik} and the task triplication decision h_i'. In addition, the task triplication decision h_i' is influenced by the task allocation decision q_{ik}. Therefore, the communication and computation energy of the nodes can be optimized by adjusting q_{ik}. The task sequence constraint (14) and the task non-preemption constraint (15) show that the value of q_{ik} is also restricted by the task start

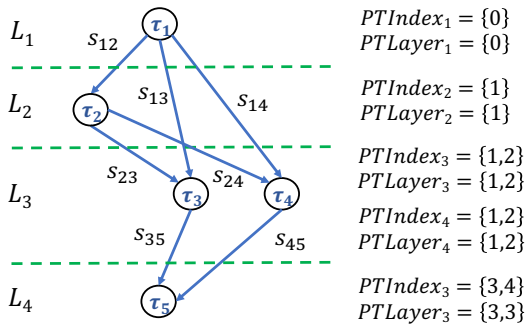


Fig. 2. Determine frequency assignment sequence based on task layer.

time t_i^s and the task sequence u_{ij}. Therefore, when making the task allocation decision q_{ik}, the constraints (14) and (15) should be taken into account.

At this step, under the given frequency assignment decision c_{il}, we aim to determine task allocation q_{ik}, task sequence u_{ij}, task triplication h'_i, and task start time t_i^s. Therefore, the Task Mapping Problem (TMP) is formulated as

$$\textbf{TMP}: \min_{\boldsymbol{q},\boldsymbol{u},\boldsymbol{h}',\boldsymbol{t}^s} \left(\max_{\forall k \in \mathcal{M}} \left\{ \left(E_k^t + E_k^d \right) / E_k^{ini} \right\} \right) \tag{38}$$
$$\text{s.t. } (3), (10) - (12), (14) - (19).$$

Since the binary variables q_{ik}, h' and u_{ij} and the continuous variable t_i^s are coupled with each other linearly, and the nonlinear items $q_{im}q_{jn}$ and $h'_i h'_j q_{im} q_{jn}$ are included in constraints (3), (17)–(19), TMP is a MINLP problem. Based on the structure of TMP, we propose Algorithms 3–4 to solve it.

Algorithm 3 first initializes the allocation index $O[i]$ for each task τ_i, where $O[i] = k$ represents that task τ_i is allocated to node θ_k. Then, it allocates tasks, one by one, to their proper nodes, causing the minimum increase in energy consumption among the nodes, under real-time and energy constraints (Lines 1–2). Different from the FAP, the allocations of dependent tasks on the wireless nodes are considered in TMP. Therefore, the real-time constraint in TMP includes the communication time of the nodes. The task allocation sequence is given by *AssSequence*, as it can handle the problems of task start time and task sequence simultaneously (i.e., t_i^s and u_{ij}). To find the maximum energy consumption at a node (i.e., $MaxCostN$), under the given task allocation decision $O[i] = k$, we design Algorithm 4. During the allocation process, if tasks τ_i and τ_j $(i \neq j)$ are dependent and these tasks are allocated to the same node, there is no need to triplicate task τ_i, since the data of τ_i is directly transmitted to τ_j within one node.

As shown in Fig. 1, τ_1, τ_4 and τ_7 are the original tasks, while τ_2, τ_3, τ_5 and τ_6 are the corresponding replicas. Based on task dependency o_{ij} and task execution cycles e_i, we follow the sequence: $AssSequence = \{\tau_1 \rightarrow \tau_2 \rightarrow \tau_3 \rightarrow \tau_4 \rightarrow \tau_5 \rightarrow \tau_6 \rightarrow \tau_7\}$ to perform task allocation. Since the original task and the corresponding replicas are independent (e.g., τ_1, τ_2 and τ_3), they can be placed in the same layer. If the tasks τ_1, τ_4, τ_5 and τ_6 are allocated to the same node, there is no need to triplicate τ_1. Hence, tasks τ_2 and τ_3 can be removed from the task graph, and the execution time of tasks τ_2 and τ_3 can be set to $t_2^c = t_3^c = 0$, since $e_2 = e_3 = 0$. Based on the allocation of the original tasks τ_4 and τ_7, we can use a similar method to decide the existence of tasks τ_5 and τ_6.

Algorithm 3: Task-to-node allocation

```
Initialize: O[i] = −1, ∀i ∈ N;
Sort tasks {τ_1, ..., τ_N} according to AssSequence;
for ∀i ∈ N do
    MinMaxE = ∞;
    for ∀k ∈ M do
        if τ_i can be allocated θ_k then
            O[i] = k;
            if τ_i is allocated to the same node as its previous dependent tasks then
                Update {t_1^e, ..., t_{i−1}^e};
            else
                Continue;
            end if
            Calculate t_i^e, E_k^t and E_k^d based on q_ik, o_ij and {t_1^e, ..., t_{i−1}^e};
            Calculate E_k^all based on E_k^t, E_k^d, E_k^s and E_k^a;
            MaxE = MaxNodeE;
            if E_k^all > E_k^l || t_i^e > H then
                q_ik = 0;
                Continue;
            else
                if MaxE < MinMaxE then
                    MinMaxE = MaxE;
                    flag = k;
                else
                    MinMaxE = MinMaxE;
                    flag = flag;
                end if
            end if
        else
            q_ik = 0;
            Continue;
        end if
    end for
    if flag == −1 then
        Fail to allocate task τ_i;
    else
        O[i] = flag;
    end if
end for
Calculate q_ik according to O[i];
```

6.3 Time Complexity

Finally, the time complexities of our frequency assignment and task mapping algorithms are discussed in terms of the number of variables [18]. The complexities of FAP and TMP are dominated by Algorithm 1 and Algorithm 3, respectively. Algorithm 2 is used to calculate the maximum energy required to execute a task, i.e., $MaxCompE$, under the given frequency assignment decision $S[i] = l$; and Algorithm 2 is used to calculate the maximum energy consumption at a node, i.e., $MaxNodeE$, under the given task allocation decision $O[i] = k$. In Algorithm 1 and Algorithm 3, the

Algorithm 4: Calculate $MaxNodeE$

```
MaxNodeE = 0;
for ∀k ∈ M do
    for ∀i ∈ N (O[i] ≠ −1) do
        Calculate E_k^t and E_k^d based on O[i] = k;
    end for
    if (E_k^t + E_k^d)/E_k^init > MaxNodeE then
        MaxNodeE = (E_k^t + E_k^d)/E_k^init;
    else
        MaxNodeE = MaxNodeE;
    end if
end for
```

variables regarding the assignment of frequency and the allocation of the task are $\boldsymbol{c} = [c_{il}]_{N\times L}$ and $\boldsymbol{q} = [q_{ik}]_{N\times M}$. Therefore, the complexities of FAP and TMP are $O(NL)$ and $O(NM)$, respectively.

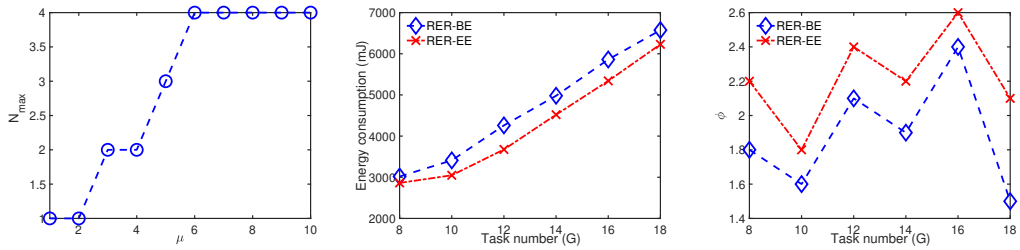
7 SIMULATION RESULTS

To evaluate the performance of the proposed approaches, we consider a networked system with 20 nodes, where 10 nodes equipped with sensors and 5 nodes connected to actuators (i.e., $M = 20$, $M_s = 10$, and $M_a = 5$). The modeling of the energy consumed by processors and wireless communication is based on [8, 41], and the characteristic of tasks is given by [42, 43], which is based on the scenario of enhancing federated learning in Fog-aided IoT.

The energy supply of node θ_k is set to $E_k^r = \eta E_k^h$, where $\eta \in [0, 1]$ is an energy efficiency factor and E_k^h is the minimum energy required to execute tasks $\{\tau_1, \ldots, \tau_N\}$. In particular, $E_k^h = E_{k,\min}^t + E_{k,\min}^d$, where $E_{k,\min}^t = N \min_{\forall m,n,k}\{r_{mnk}\}$ is the minimum energy required to transmit all the task data, while $E_{k,\min}^d = \sum_{i\in\mathcal{N}}(\min_{\forall l} \frac{e_i}{f_l} P_l^c)$ is the minimum energy required to execute all the tasks. Since the nodes equipped with the sensors and the actuators consume more energy for executing sensing and control tasks, we set $E_k^r = 2\eta E_k^h$ for the sensor nodes while $E_k^r = 3\eta E_k^h$ for the actuator nodes.

The scheduling horizon (i.e., task deadline) is set to $H = \max_{\forall j}\{d_j\}$, where $d_j = \max_{\forall i, o_{ij}=1}\{d_i\} + \hat{t}_j^r + \hat{d}_j$. $\hat{t}_j^r$ and $\hat{d}_j$ are the temporary data receiving time and the relative deadline of task τ_j (the bound of task execution time), respectively. $\hat{t}_j^r$ is assumed within the range $[N \min_{\forall m,n}\{t_{mn}\}, N \max_{\forall m,n}\{t_{mn}\}]$, where $N \min_{\forall m,n}\{t_{mn}\}$ and $N \max_{\forall m,n}\{t_{mn}\}$ are the minimum and the maximum time required to transmit the data of N tasks between two nodes, respectively. Similarly, $\hat{d}_j$ is assumed within the range $[\min_{\forall l}\{\frac{e_i}{f_l}\}, \max_{\forall l}\{\frac{e_i}{f_l}\}]$, where $\min_{\forall l}\{\frac{e_i}{f_l}\}$ and $\max_{\forall l}\{\frac{e_i}{f_l}\}$ are the minimum and the maximum time required to execute a task with e_i cycles, respectively.

First, we show the influence of system parameters on the problem solution (e.g., task allocation and the value of the objective function). Then, we compare the system performance (i.e., the energy consumption of the nodes, the reliability of the system, and the schedulability of the problem) of the proposed task mapping scheme (Reliability-Energy-Realtime – RER) (i.e., PP) and other state-of-the-art mapping schemes (i.e., Energy-Efficiency (EE) [35], Task Duplication (TD) [15], and Task Triplication (TT) [19]). Finally, we evaluate the algorithm performance (i.e., the solution quality, the computation time, and the problem feasibility) of the proposed ORTM and HRTM algorithms with the Branch and Bound (B&B) [5], which can find the optimal solution for MILP problems, and the heuristics based on As-Soon-As-Possible (ASAP) [21]. The ASAP method estimates the as-soon-as-possible start time and uses it to limit the range of the task start time.



(a) The influence of μ on task allocation decision q_{ik}.

(b) Total node energy consumption with RER-BE and RER-ME.

(c) Node energy consumption gap with RER-BE and RER-ME.

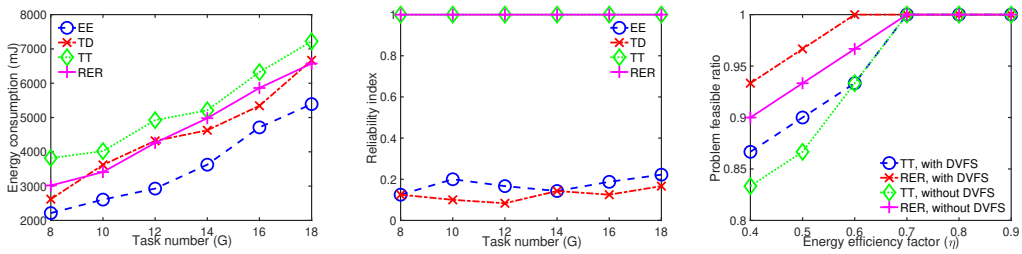
Fig. 3. System performance comparison under different parameters.

The simulations are performed on a laptop with a quad-core 2.5 GHz Intel I7 processor and 16 GB RAM, and the algorithms are implemented in Matlab 2020b. Note that different processor platforms and task graphs will lead to different parameters for the PP. However, the problem structures under different parameters are the same; thus, the proposed methods are still applicable. In addition, different algorithms can be compared under the given system parameters.

Fig. 3(a) explores the influence of energy ratio between communication and computation on task allocation decision q_{ik}. We set $G = 12$, $\eta = 0.8$, and change the value of μ from 1 to 10 with a step of 1. Let $\mu = e_{\max}^{comm}/e_{\max}^{comp}$ and $N_{\max} = \max_{\forall k}\{N_k\}$, where $e_{\max}^{comm} = \max_{\forall \beta,\gamma,k}\{e_{\beta\gamma k}\}$ is the maximum energy required for a node to relay task data, and $e_{\max}^{comp} = \max_{\forall i,l}\{\frac{e_i}{f_l}P_l\}$ is the maximum energy required for a node to execute a task. Therefore, the larger the value of μ is, the more energy is required to transmit task data, compared with the energy consumed for task execution. Since N_k is the number of tasks allocated to node θ_k, $N_{\max} = \max_{\forall k}\{N_k\}$ represents the maximum number of tasks assigned to a node. Fig. 3(a) shows that $N_{\max}$ increases with μ. This is because, with larger communication energy compared to the computation energy, the trend is allocating dependent tasks to the same node to reduce the communication energy.

Fig. 3(b) and Fig. 3(c) compare the energy consumption of the nodes with two task mapping aims: 1) the PP with an objective function to Balance node Energy consumption (RER-BE), i.e., $\min[\max_{\forall k}\{(E_k^t+E_k^d)/E_k^{ini}\}]$, and 2) the PP with an objective function to Minimize total node Energy consumption (RER-ME), i.e., $\min[\sum_{k\in\mathcal{M}}(E_k^t + E_k^d)]$. To evaluate the performances of RER-BE and RER-ME, we introduce an index $\phi = (\max_{\forall k}\{E_k^t+E_k^d\})/(\min_{\forall k}\{E_k^t+E_k^d\})$, where $E_k^t+E_k^d \neq 0$. Since $E_k^t+E_k^d$ is the computation and the communication energy of node θ_k, ϕ is the energy consumption gap between the nodes. The smaller the value of ϕ is, the better balance of node energy consumption can be achieved. We set $\eta = 0.8$ and change the value of G (the number of original tasks) from 8 to 18 with a step of 2. From Fig. 3(b), we observe that the total node energy consumption of RER-ME is lower than RER-BE. This is because by using RER-ME, the dependent tasks tend to be allocated to the same nodes, so as to reduce the communication energy of the nodes. On the other hand, Fig. 3(c) shows that the value of ϕ with RER-BE is lower than RER-ME. Hence, by using RER-BE, we can avoid some nodes depleting energy early, i.e., enhancing the lifetime of the networked system.

Fig. 4(a) compares the energy consumption of the nodes with EE, TD, TT, and the proposed RER scheme. The aim of the EE scheme is to minimize the energy consumption of the nodes under real-time and energy constraints. With the TD scheme, each task is duplicated, while with the TT scheme each task is triplicated. We set $\eta = 0.8$ and change the value of G from 8 to 18 with a step of 2. From Fig. 4(a), we observe that for all mapping schemes, with the task number G increasing, the energy consumption of the nodes increases as well, since more energy is required to execute



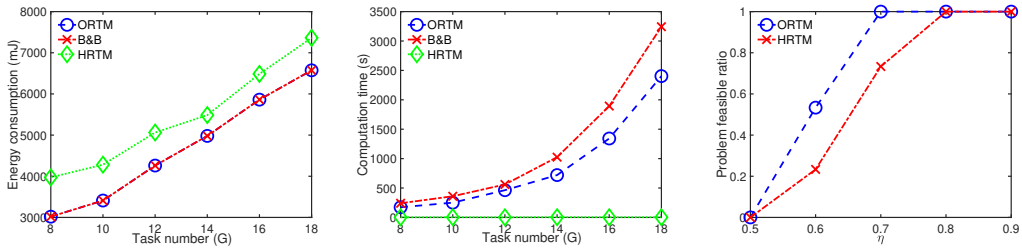
(a) Energy consumption of different task mapping schemes with N varying. (b) Reliability index of different task mapping schemes with N varying. (c) Schedulability of different task mapping schemes with η varying.

Fig. 4. System performance comparison with different task mapping schemes.

the tasks and exchange the task data between the nodes. In addition, the simulation results show that under the given task number G, the EE scheme consumes the least energy, while the TT scheme consumes more energy than the other schemes. This is because EE aims to minimize the energy consumption of the nodes, and the tasks are not duplicated/triplicated. However, each task is triplicated in TT, which leads to more tasks being required to execute. Fig. 4(a) also shows that although the approaches achieve a similar reliability level, RER consumes less energy consumption than TT (28.62% on average), since only parts of tasks are triplicated in RER.

Fig. 4(b) compares the probability of original tasks against the False Data Injection (FDI) under EE, TD, TT, and RER schemes. We introduce a reliability index $\sum_{i=1}^{G} w_{\alpha_i}/G$ for each task mapping scheme. With the EE scheme, since the tasks are not replicated, we have $N = G$ (G and N are the numbers of original tasks and total tasks, respectively), while with TD, TT, and RER schemes, we get $N = 2G$, $N = 3G$ and $N = 3G$, respectively. On the other hand, the value of parameter w_{α_i} is determined as follows: 1) If the original task τ_{α_i} is an entry task since τ_{α_i} will not receive data from other tasks, we set $w_{\alpha_i} = 1$; 2) If the original task τ_{α_i} is a non-entry task, and this task receives three copies (i.e., the original, duplicated and triplicated tasks) from each previous dependent task or just one copy (i.e., the original task) from each previous dependent task, which is assigned to the same node as task τ_{α_i}, we set $w_{\alpha_i} = 1$, else, we set $w_{\alpha_i} = 0$. From Fig. 4(b), we observe that under the same value of G, the reliability indexes of TT and RER are higher than EE and TD, since compared with EE and TD, the tasks are triplicated in TT and RER. In addition, because the tasks are not triplicated in EE and TD, the reliability index of TD is not always higher than EE. Although the reliability indexes of TT and RER are the same, with TT the nodes consume more energy, as more tasks are triplicated.

Fig. 4(c) evaluates the schedulability of TT and RER schemes since both of them perform task triplication. To compare their schedulabilities, the problem feasible ratio is introduced as a metric. For TT and RER schemes, we construct and solve the corresponding task mapping problem 30 times ($n_a = 30$), each time with fixed parameters G and η, and tuned parameters o_{ij} and e_i. Let n_t and n_r denote the times that the task mapping problems in TT and RER are feasible, respectively. Hence, the problem feasible ratios for TT and RER schemes can be defined as n_r/n_a and n_r/n_a, respectively. We set $G = 14$ and change the value of η from 0.4 to 0.9 with a step of 0.1. From Fig. 4(c), we observe that with η increasing, the problem feasible ratios of TT and RER both increase. In addition, the scheme with DVFS has a higher problem feasible ratio than the scheme without DVFS. This is because by introducing the frequency-to-task assignment variable c_{il} into the task mapping problem, the time and the energy regarding task execution can be further optimized. Hence, the real-time and energy constraints can be easier satisfied, especially when the energy factor η is small. Fig. 4(c) also shows that under the given G and η, RER has a higher problem feasible ratio



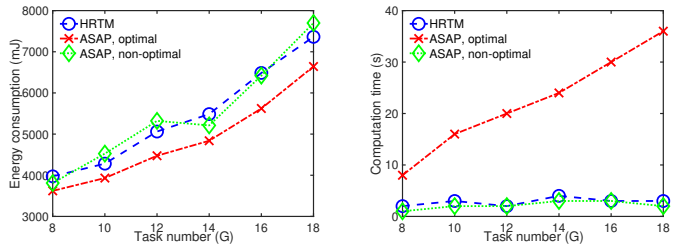
(a) Energy consumption of ORTM, HRTM and B&B with G varying. (b) Computation time of ORTM, HRTM and B&B with G varying. (c) Feasibility of task mapping problem with ORTM and HRTM.

Fig. 5. Algorithm performance comparison of ORTM, HRTM and B&B.

than TT. This is because not all original tasks are triplicated in RER, and thus, the nodes in RER consume less energy for task execution and data exchange than the nodes in TT. From Fig. 4, we can observe that compared with EE, TD, and TT schemes, the proposed RER scheme makes better use of system resources to enhance system reliability.

The solution quality and the computation time of using B&B and the proposed ORTM and HRTM algorithms to solve the PP under different task numbers G are compared in Fig. 5(a) and Fig. 5(b). The results show that the solutions found by ORTM and B&B are the same, i.e., the ORTM algorithm is also optimal. In addition, the ORTM algorithm achieves a lower objective function value than that of the HRTM algorithm (26.35% on average). With task number G increasing, more variables and constraints are added to the PP. Therefore, the computation time required by B&B and ORTM both increases. However, the computation time of ORTM is lower than that of B&B (24.84% on average). This is because the computational complexity of an optimization problem is highly related to the number of variables and constraints. Solving the smaller problems with fewer variables and constraints iteratively is more efficient than solving a single large problem [37]. In addition, compared with the ORTM, the HRTM has a negligible computation time, since this method only needs to solve two polynomial-time problems FAP and TMP in sequence. The schedulabilities of ORTM and HRTM algorithms to solve task mapping problem with the RER scheme are compared in Fig. 5(c). We use problem feasible ratio as a metric, and we set $n_a = 30$, $G = 14$, and $\eta \in [0.3, 0.8]$. Fig. 5(c) shows that the problem feasible ratios of ORTM and HRTM increase with η. Since the bounds of the constraints can be relaxed by η, the feasible region of the task mapping problem is enlarged. However, the problem feasibility ratio of ORTM is always higher than HRTM. This is because ORTM optimizes multiple variables at the same time, while HRTM optimizes these variables step by step.

Fig. 6(a) and Fig. 6(b) compare the solution quality and the computation time of HRTM and ASAP-based heuristic used to solve the PP. To solve this problem, the difficulty lies in the way of dealing with the continuous variable t_i^s (task start time). This is because when the value of t_i^s is fixed, the PP will reduce to an integer programming problem that contains only the binary variables q_{ik}, c_{il}, u_{ij} and h'_i. This problem is easier to solve than the MILP problem with continuous and binary variables. To perform the ASAP-based heuristic, in the first step, we assume that the start time of the entry task is 0. Then, for the adjacent dependent tasks, e.g., τ_i and τ_j ($o_{ij} = 1$), we add a temporary communication time θ_{ij} among them. The value of θ_{ij} is set to $(\max_{\forall m,n}\{t_{mn}\} + \min_{\forall m,n}\{t_{mn}\})/2$, so as to model the communication time when tasks τ_i and τ_j are assigned to the different nodes. Therefore, the start time of task τ_j can be replaced by $t_j^s = t_i^e + \theta_{ij}$. For example, as the task graph illustrated in Fig. 2, we have $t_1^s = 0$, $t_2^s = t_1^e + \theta_{12} = \sum_{l \in \mathcal{L}} c_{1l} \frac{h'_1 e_1}{f_l} + \theta_{12}$, $t_3^s = t_1^e + \theta_{13}$, $t_4^s = t_1^e + \theta_{14}$ and



(a) Energy consumption of HRTM and ASAP with G varying.

(b) Computation time of HRTM and ASAP with G varying.

Fig. 6. Algorithm performance comparison of HRTM and ASAP.

$t_5^s = t_3^e + \theta_{35} = t_3^s + \sum_{l \in \mathcal{L}} c_{3l} \frac{h'_3 e_3}{f_l} + \theta_{35}$. By replacing continuous variable t_i^s with binary variables c_{il} and h'_i, we next solve an integer programming problem with binary variables q_{ik}, c_{il}, u_{ij} and h'_i. Finally, when the solution is found, the temporary communication time θ_{ij} is updated by q_{ik}, and the task start time t_i^s is updated by h'_i and θ_{ij}.

From the above statement, we can see that the computation time of the ASAP-based heuristic is mainly dominated by solving the integer programming problem. To solve this problem, we can use either the optimal method (e.g., B&B) or the heuristic method (e.g., Feasibility Pump (FP) [14]). Fig. 6(a) and Fig. 6(b) show that when using the optimal method to solve the integer programming problem in ASAP, the computation time of HRTM is lower than ASAP. However, ASAP achieve a higher solution quality than HRTM. This is because the binary variables q_{ik}, c_{il}, u_{ij}, and h'_i are solved simultaneously in ASAP, while they are solved sequentially in HRTM, according to two subproblems FAP and TMP. The variable c_{il} is first determined in FAP, and then, the variables q_{ik}, u_{ij} and h'_i are determined in TMP. In addition, the binary variables q_{ik}, c_{il} and h'_i are coupled with each other nonlinearly in ASAP. To deal with the nonlinear items, more auxiliary variables and additional constraints are involved in ASAP. Hence, compared with HRTM, ASAP has a longer computation time. On the other hand, when using the heuristic approach (e.g., FP) to solve the integer programming problem in ASAP, the solution quality and the computation time both decrease, since the FP only provides a feasible solution. In that case, the gap between the computation time (the solution quality) of HRTM and ASAP is small, compared with using the optimal method to solve the integer programming problem in ASAP.

8 CONCLUSION

In this paper, we study the task mapping problem on the wireless nodes of networked systems. To enhance the reliability of data transmission among the nodes, as well as to satisfy the real-time and energy constraints, we introduce the task triplication and the DVFS scheme into the task mapping problem. The aim is to decide task-to-node allocation, frequency-to-task assignment, task triplication, and task scheduling so that the energy consumption of the nodes is balanced. The problem is first formulated as MINLP, then transformed into an MILP by adding auxiliary variables and additional constraints. Based on the structure of the problem, we propose an ORTM algorithm to find the optimal solution. In addition, to deal with large network sizes, we also design an HRTM algorithm with reduced computation time. The simulation results show that the proposed reliability-energy-time aware task mapping scheme outperforms other schemes regarding transmission reliability and energy efficiency. In addition, the solution found by ORTM is guaranteed to converge to the optimal solution, while the HRTM is able to find a proper solution within a negligible computation time compared with ORTM.

9 ACKNOWLEDGEMENT

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFF0902800, in part by the National Natural Science Foundation of China under Grant 92067111, in part by the Fundamental Research Funds for the Central Universities of China under Grants 2242021R10113 and 2242022K30038, in part by the Southeast University "Zhishan Scholars" Projects under Grant 2242021R40003, in part by Jiangsu Province Frontier Leading Project under Grant BK20202006, and in part by the Jiangsu Provincial Key Research and Development Program under Grant BE2022135.

REFERENCES

[1] M. Aazam, S. Zeadally, and K. A. Harras. 2018. Deploying fog computing in industrial Internet of things and industry 4.0. *IEEE Trans. Ind. Informat.* 14, 10 (2018), 4674–4682.

[2] X. Bai, L. Liu, M. Cao, J. Panneerselvam, Q. Sun, and H. Wang. 2017. Collaborative actuation of wireless sensor and actuator networks for the agriculture industry. *IEEE Access* 5 (2017), 13286–13296.

[3] J. F. Benders. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.* 4, 1 (1962), 238–252.

[4] B. Billet and V. Issarny. 2014. From task graphs to concrete actions: a new task mapping algorithm for the future Internet of Things. In *Proc. IEEE International Conference on Mobile Ad Hoc and Sensor Systems.* 470–478.

[5] S. Boyd and J. Mattingley. 2007. Branch and bound methods. *Notes for EE364b, Stanford University* (2007), 1–11.

[6] S. Boyd and L. Vandenberghe. 2004. Convex optimization. *Cambridge University Press* (2004).

[7] S. Burer and A. N. Letchford. 2012. Non-convex mixed-integer nonlinear programming: a survey. *Surveys in Oper. Res. Manag. Sci.* 17, 2 (2012), 97 – 106.

[8] G. Chen, K. Huang, and A. Knoll. 2014. Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM Combination. *ACM Trans. Embed. Comput. Syst.* 13, 3 (2014), 111:1–111:21.

[9] Y. Chen, S. Huang, F. Liu, Z. Wang, and X. Sun. 2019. Evaluation of reinforcement learning-based false data injection attack to automatic voltage control. *IEEE Trans. Smart Grid* 10, 2 (2019), 2158–2169.

[10] M. Cui, A. Kritikakou, L. Mo, and E. Casseau. 2021. Fault-tolerant mapping of real-time parallel applications under multiple DVFS schemes. In *IEEE Real-Time and Embedded Technology and Applications Symposium.* 387–399.

[11] M. Cui, A. Kritikakou, L. Mo, and E. Casseau. 2022. Energy-efficient partial-duplication task mapping under multiple DVFS schemes. *Int. J. Parallel. Prog.* 50, 1 (2022), 267–294.

[12] N. Edalat, C. K. Tham, and W. Xiao. 2012. An auction-based strategy for distributed task allocation in wireless sensor networks. *Computer Communications* 35, 8 (2012), 916 – 928.

[13] A. Emeretlis, G. Theodoridis, P. Alefragis, and N. Voros. 2017. Static mapping of applications on heterogeneous multi-core platforms combining logic-based Benders decomposition with integer linear programming. *ACM Trans. Des. Autom. Electron. Syst.* 23, 2 (2017), 26:1–26:24.

[14] M. Fischetti, F. Glover, and A. Lodi. 2005. The feasibility pump. *Math. Program.* 104, 1 (2005), 91–104.

[15] C. Gou, A. Benoit, M. Chen, L. Marchal, and T. Wei. 2018. Reliability-aware energy optimization for throughput-constrained applications on MPSoC. In *Proc. IEEE International Conference on Parallel and Distributed Systems.* 1–10.

[16] D. Guo, S. Gu, J. Xie, L. Luo, X. Luo, and Y. Chen. 2021. A mobile-assisted edge computing framework for emerging IoT applications. *ACM Trans. Sen. Netw.* 17, 4 (2021).

[17] M. A. Haque, H. Aydin, and D. Zhu. 2017. On reliability management of energy-aware real-time systems through task replication. *IEEE Trans. Parallel Distrib. Syst.* 28, 3 (2017), 813–825.

[18] O. He, S. Dong, W. Jang, J. Bian, and David Z. Pan. 2012. UNISM: Unified scheduling and mapping for general networks on chip. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 20, 8 (2012), 1496–1509.

[19] S. Kang, H. Yang, S. Kim, I. Bacivarov, S. Ha, and L. Thiele. 2014. Reliability-aware mapping optimization of multi-core systems with mixed-criticality. In *Proc. IEEE Design, Automation Test in Europe Conference Exhibition.* 1–4.

[20] S. K. Khaitan and J. D. McCalley. 2015. Design techniques and applications of cyber-physical systems: a survey. *IEEE Syst. J.* 9, 2 (2015), 350–365.

[21] L. F. Leung, C. Y. Tsui, and W. H. Ki. 2003. Simultaneous task allocation, scheduling and voltage assignment for multiple-processors-core systems using mixed integer nonlinear programming. In *Proc. IEEE International Symposium on Circuits and Systems.* 309–312.

[22] D. Li and J. Wu. 2015. Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms. *IEEE Trans. Parallel Distrib. Syst.* 26, 3 (2015), 810–823.

[23] H. Li, L. Lai, and H. V. Poor. 2012. Multicast routing for decentralized control of cyber physical systems with an application in smart grid. *IEEE J. Sel. Areas Commun.* 30, 6 (2012), 1097–1107.

[24] W. Li, F. C. Delicato, P. F. Pires, Y. C. Lee, A. Y. Zomaya, C. Miceli, and L. Pirmez. 2014. Efficient allocation of resources in multiple heterogeneous wireless sensor networks. *J. Parallel and Distrib. Comput.* 74, 1 (2014), 1775 – 1788.
[25] D. Lin, Q. Wang, W. Min, J. Xu, and Z. Zhang. 2020. A survey on energy-efficient strategies in static wireless sensor networks. *ACM Trans. Sen. Netw.* 17, 1 (2020).
[26] J. Liu, C. Bondiombouy, L. Mo, and P. Valduriez. 2022. Two-phase scheduling for efficient vehicle sharing. *IEEE Trans. Intell. Transp. Syst.* 23, 1 (2022), 457–470.
[27] J. Liu, L. Mo, S. Yang, J. Zhou, S. Ji, H. Xiong, and D. Dou. 2021. Data placement for multi-tenant data federation on the cloud. *IEEE Trans. Cloud Comput.* (2021), 1–16. https://doi.org/10.1109/TCC.2021.3136577
[28] K. Manandhar, X. Cao, F. Hu, and Y. Liu. 2014. Detection of faults and attacks including false data injection attack in smart grid using Kalman filter. *IEEE Trans. Control Netw. Syst.* 1, 4 (2014), 370–379.
[29] L. Mo, A. Kritikakou, and O. Sentieys. 2018. Controllable QoS for imprecise computation tasks on DVFS multicores with time and energy constraints. *IEEE J. Emerg. Sel. Topic Circuits Syst.* 8, 4 (2018), 708–721.
[30] L. Mo, P. You, X. Cao, Y. Song, and A. Kritikakou. 2019. Event-driven joint mobile actuators scheduling and control in cyber-physical systems. *IEEE Trans. Ind. Informat.* 15, 11 (2019), 5877–5891.
[31] Y. Mo, R. Chabukswar, and B. Sinopoli. 2014. Detecting integrity attacks on SCADA systems. *IEEE Trans. Control Syst. Technol.* 22, 4 (2014), 1396–1407.
[32] Y. Mo, T. H. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli. 2012. Cyber-physical security of a smart grid infrastructure. *Proc. IEEE* 100, 1 (2012), 195–209.
[33] G. R. Mode, P. Calyam, and K. A. Hoque. 2020. Impact of false data injection attacks on deep learning enabled predictive analytics. *Proc. IEEE/IFIP Network Operations and Management Symposium* (2020), 1–7.
[34] K. Ota, M. Dong, Z. Cheng, J. Wang, X. Li, and X. Shen. 2012. ORACLE: mobility control in wireless sensor and actor networks. *Comput. Commun.* 35, 9 (2012), 1029–1037.
[35] A. Pathak and V. K. Prasanna. 2010. Energy-efficient task mapping for data-driven sensor network macroprogramming. *IEEE Trans. Comput.* 59, 7 (2010), 955–968.
[36] L. P. Qian, Y. J. Zhang, Y. Wu, and J. Chen. 2013. Joint base station association and power control via Benders' decomposition. *IEEE Trans. Wireless Commun.* 12, 4 (2013), 1651–1665.
[37] C. D. Randazzo and H. P. L. Luna. 2001. A comparison of optimal methods for local access uncapacitated network design. *Ann. Oper. Res.* 106, 1 (2001), 263–286.
[38] G. Raravi and V. Nélis. 2014. Task assignment algorithms for heterogeneous multiprocessors. *ACM Trans. Embed. Comput. Syst.* 13, 5 (2014), 159:1–159:26.
[39] A. Sikder, G. Petracca, H. Aksu, T. Jaeger, and A. Uluagac. 2018. A survey on sensor-based threats to Internet-of-Things (IoT) devices and applications. arXiv:1802.02041.
[40] U. Tariq, H. Ali, L. Liu, J. Hardy, M. Kazim, and W. Ahmed. 2021. Energy-aware scheduling of streaming applications on edge-devices in IoT-based healthcare. *IEEE trans. green commun. netw.* 5, 2 (2021), 803–815.
[41] Y. Tian and E. Ekici. 2007. Cross-layer collaborative in-network processing in multihop wireless sensor networks. *IEEE Trans. Mobile Comput.* 6, 3 (2007), 297–310.
[42] G. Xie, Y. Chen, X. Xiao, C. Xu, R. Li, and K. Li. 2018. Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems. *IEEE Trans. Sustain. Comput.* 3, 3 (2018), 167–181.
[43] J. Yao and N. Ansari. 2021. Enhancing federated learning in fog-aided IoT by CPU frequency and wireless power control. *IEEE Internet Things J.* 8, 5 (2021), 3438–3445.
[44] D. Ye and T. Zhang. 2020. Summation detector for false data-injection attack in cyber-physical systems. *IEEE Trans. Cybern.* 50, 6 (2020), 2338–2345.
[45] L. Yeh, C. Lu, C. Kou, Y. Tseng, and C. Yi. 2010. Autonomous light control by wireless sensor and actuator networks. *IEEE Sensors J.* 10, 6 (2010), 1029–1041.
[46] L. Zhang, K. Li, and C. Li. 2017. Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems. *Information Sciences* 379 (2017), 241–256.