Research Repository

Energy-Aware Task Mapping Combining DVFS and Task Duplication for Multicore Networked Systems

Energy-Aware Task Mapping Combining DVFS and Task Duplication for Multicore Networked Systems

Lei Mo[a,*], Jingyi Zhang[a], Minyu Cui[b], Xiaoyong Yan[c], Shuang Wang[d] and Xiaojun Zhai[e]

[a]*Key Laboratory of Measurement and Control of CSE, Ministry of Education, School of Automation, Southeast University, Nanjing, 210096, China*
[b]*Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, 412 96, Sweden*
[c]*School of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing, 210003, China*
[d]*School of Computer Science and Engineering, Southeast University, Nanjing, 211189, China*
[e]*School of Computer Science and Electronic Engineering, University of Essex, CO4 3SQ, Colchester, U.K.*

ARTICLE INFO

Keywords:
Multicore embedded systems
Dependent and real-time tasks
Task mapping
Task reliability

ABSTRACT

Integrating high-performance communication and computation capabilities, multicore embedded platforms have become key components to realize applications of networked systems, e.g., Cyber-Physical Systems (CPS). Such systems usually consist of multiple dependent and real-time tasks that can be executed in parallel on different cores of the nodes and have timing, energy, and reliability constraints. Designing efficient task mapping methods to transmit and process task data under multiple constraints is challenging. Existing works seldom consider the joint design problem under timing, energy, and reliability constraints, which are coupled with each other, introducing complexity in designing efficient task mapping methods. In this paper, we first formulate the joint design problem as a complex combinational optimization problem and design a linearization method to find the optimal solution. To reduce computation complexity and enhance scalability, we design a decomposition-based heuristic method. Then, a refinement method based on feedback control is added to enhance task schedulability. The results show that the optimal solution obtained by the proposed method achieves the desired system performance. Moreover, the proposed heuristic provides a feasible solution with negligible computing time (reduces 99.9% computation time but with 24.3% performance loss). Compared with the existing works, our method can optimize the usage of system resources to balance energy, timing, and reliability requirements.

1. Introduction

Common applications of Cyber-Physical Systems (CPS) include sensing, control, data transmission and processing, under real-time and data dependency constraints [27]. These applications are typically modeled by a set of Directed Acyclic Graph (DAG) dependent tasks [29, 8]. Due to the need for energy and computation efficiency of CPS nodes, multicores have become promising architectures for networked embedded systems [3, 12]. Multicore architectures can process several tasks simultaneously to improve task execution efficiency, compared with single-core ones, where the tasks are processed sequentially. Besides data processing capabilities, the CPS embedded systems typically integrate wireless communication capabilities [21, 34]. Since communication and computation tasks are performed iteratively on the CPS nodes (e.g., sensors, actuators, or controllers) [8, 32], parallel processing is possible not only among different CPS nodes but also inside a CPS node, when multicore architectures are used. To meet the desired system requirements, application tasks should be mapped appropriately (i.e., task allocation and scheduling) on the CPS nodes and their processing elements.

Timing and energy are critical metrics for evaluating the system performance during the task mapping process [31, 6]. If the real-time tasks are not finished within the deadlines, this can lead to errors with potentially catastrophic consequences. Energy consumption is vital for networked embedded systems, especially for battery-powered or energy-harvesting devices. However, energy efficiency and real-time response usually contradict each other. To address the energy efficiency and real-time response trade-off, Dynamic Voltage and Frequency Scaling (DVFS) is used [17, 10], as it changes the working frequency and supply voltage of the processing elements of the embedded system during the task execution process. To reduce energy consumption, the voltage/frequency can be lower, which makes the task execution time longer. However, it also decreases task reliability since the transient failure rate is increased [36, 7]. Task replication can improve reliability [36, 19], e.g., task duplication. Therefore, task reliability, energy efficiency, and real-time response should be jointly optimized when mapping real-time tasks on processing elements and nodes of networked systems under multiple constraints. Furthermore, the CPS nodes are usually connected through a mesh network [21, 3], where one node can transmit data to another through multiple paths determined by the network topology. Path selection defines the route of the data through the networked nodes,

*Corresponding author

lmo@seu.edn.cn (L. Mo); 220221990@seu.edn.cn (J. Zhang); minyu@chalmers.se (M. Cui); xiaoyong_yan@126.com (X. Yan); shuangwang@seu.edn.cn (S. Wang); xzhai@essex.ac.uk (X. Zhai)

ORCID(s): 0000-0002-1119-7617 (L. Mo); 0009-0004-9061-5267 (J. Zhang); 0000-0002-5983-1648 (M. Cui); 0000-0002-8097-9268 (X. Yan); 0000-0003-3405-5942 (S. Wang); 0000-0002-1030-8311 (X. Zhai)

[1]This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFF0902800 and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20242028.

i.e., it impacts the allocation and scheduling of the tasks on the nodes. Therefore, task communication and computation costs are determined by data path and task mapping, which are coupled with each other.

Task mapping works exist for multicore architectures, dealing with energy efficiency, real-time response, and task reliability, targeting a single platform (node), without considering the communication among the nodes [31, 14, 6, 36, 7]. Usually, the cores inside the platform are connected by a high-speed data bus. The communication costs (e.g., communication time or energy) among the cores are very small compared with the computation costs, and they can be omitted. When the cores are connected with a Network-on-Chip (NoC) [1, 19], the routers are responsible for the data transmission among the cores. Since the routers are usually arranged in a grid network, the XY routing can be employed for data transmission [1, 20]. Other works perform task mapping on the different nodes of networked systems to improve system performances [17, 21, 34, 16, 37], such as energy efficiency or real-time response. However, the multicore platform, task reliability, and multipath data routing are not comprehensively considered. More details are in Section 6.

To address these limitations, we target the problem of mapping a set of dependent and real-time tasks on networked nodes, realized with platforms having multiple cores and DVFS, which collaborate to execute the tasks energy-efficiently under multiple constraints. The main contributions are:

1. A Mixed-Integer Non-Linear Programming (MINLP) formulation of the task mapping problem on networked nodes, realized with multicore DVFS-enhanced platforms, under real-time, reliability, dependency, and multipath data routing constraints. The task-to-node and task-to-core allocation, frequency-to-task assignment, task duplication, and data path selection are optimized simultaneously. To find the optimal solution, we equivalently transform the original MINLP to a Mixed-Integer Linear Programming (MILP) by introducing auxiliary variables and additional constraints that replace the nonlinear terms and do not lead to solution quality degradation.
2. To improve scalability, we propose a novel heuristic method based on the complex joint-design problem structure and the principles of decomposition and feedback loop. The original problem is divided into two subproblems: Frequency Assignment and Task Duplication (FATD) and Task Allocation and Path Selection (TAPS). These subproblems are solved by low computational complexity algorithms one after the other in an iterative way. To enhance schedulability, a refinement method based on the voltage/frequency adjustment is introduced during the iteration between the two subproblems.
3. Extensive experiments are conducted to evaluate the proposed methods. The results show that our task mapping method outperforms existing methods regarding task reliability, task schedulability, and energy efficiency, as DVFS, task duplication, and multipath data routing are optimized simultaneously. Furthermore, the proposed heuristic reduces 99.9% of the computation time, with 24.3% performance loss, compared with the optimal method, being suitable for large systems.

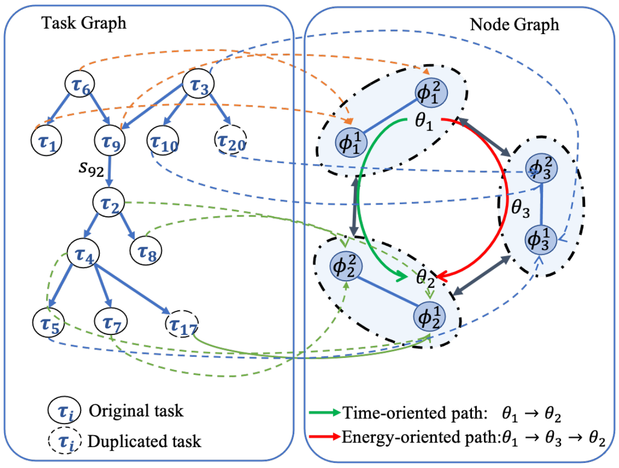


Figure 1: Motivation example: task and node graphs.

The remaining paper is organized as follows. Section 2 motivates the proposed method through an example. Section 3 introduces the system model and mathematically formulates the task mapping problem. Section 4 presents the proposed heuristic method. Section 5 shows the evaluation results. Section 6 discusses the related work, and Section 7 concludes this paper.

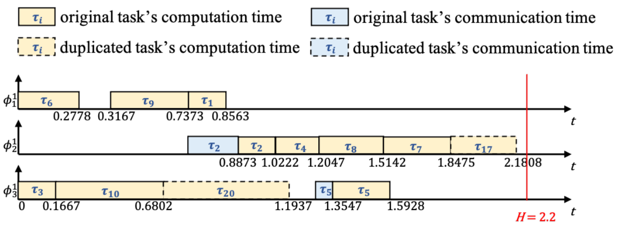
2. Motivation Example

To illustrate the problem under study and motivate the proposed method, we will use the example of Fig. 1 and Fig. 2. The original task graph is composed of $M = 10$ dependent and real-time tasks $\{\tau_1, \ldots, \tau_{10}\}$. Let τ_i and τ_{i+M} denote the original and duplicated tasks. As we combine task duplication and DVFS, only a subset of the tasks is duplicated, depending on the reliability constraints. The original and duplicated tasks are executed on a networked system with $N = 3$ nodes $\{\theta_1, \ldots, \theta_3\}$, where each node θ_k has $R = 2$ cores ϕ_k^1 and ϕ_k^2. The number of task execution cycles is set within the range $[4 \times 10^7, 6 \times 10^8]$ [31]. The scheduling horizon $H = 2.2\ s$ and the reliability constraint is given by the threshold $R_{th} = 0.998$ [7]. Fig. 1 shows the allocation of tasks on the cores and the nodes (dashed arrows), and Fig. 2 compares the scheduling of tasks for the single-core and multi-core scenarios.

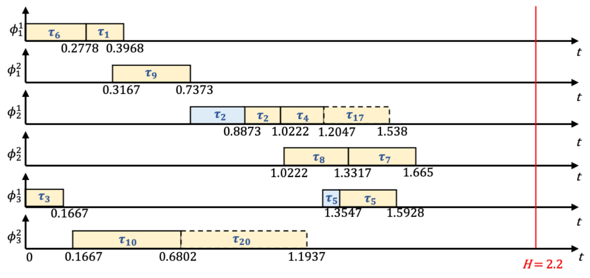
Considering a single-core platform to realize the networked system nodes, the tasks have to be executed in sequence, e.g., although tasks τ_9 and τ_1 are independent, they have to be executed in sequence on node θ_1, increasing the end time of task τ_1 to 0.8563 s. However, considering a multi-core platform, tasks τ_1 and τ_9 can be executed on different cores of θ_1. As τ_1 and τ_9 are independent, θ_1 can process these tasks simultaneously, and thus, the end time of task τ_1 is reduced to 0.7373 s. Therefore, multicore architectures are promising for enhancing the real-time response of networked systems. Note that task makespan is highly related to the optimization of task mapping decisions, which is the main objective of this paper.

As the example shown in Fig. 1, tasks τ_2 and τ_9 are dependent and allocated to nodes θ_2 and θ_1, respectively. To execute τ_2, task data is transmitted from θ_1 to θ_2. Through the network topology, there exist two data paths from θ_1 to θ_2: $\theta_1 \rightarrow \theta_2$ with communication time 0.15 s and energy 312 mJ, and $\theta_1 \rightarrow \theta_3 \rightarrow \theta_2$ with communication time 0.2 s and energy 210 mJ. If we select the energy-oriented path $\theta_1 \rightarrow \theta_3 \rightarrow \theta_2$, although it has a lower communication energy cost, the task real-time constraint is hard to satisfy, as $\max\{t_i^e\} = 2.2308\ s > H = 2.2\ s$. However, if we select the time-oriented path $\theta_1 \rightarrow \theta_2$, although the communication energy increases, all the tasks can be finished by the task deadline. Therefore, path selection is crucial and should be jointly considered during task mapping.

On the one hand, using only DVFS to meet the reliability constraints requires a high V/F level to execute the tasks, leading to a high energy consumption of 1493.2 mJ. On the other hand, using only task duplication, the time required to complete the original and their duplicated tasks is high, e.g., 3.0431 s in this example. By combining task duplication and DVFS technologies, only tasks τ_7 and τ_{10} are duplicated (denoted them as τ_{17} and τ_{20}, respectively). As the energy required for task execution has a quadratic relationship with the V/F level, executing the original and duplicated tasks with a low V/F level is more efficient than executing the original task with a high V/F level. The joint design reduces the energy consumption to 1478.6 mJ and the execution time to 2.18 s. Since fewer tasks require duplication, the real-time constraint $\max_{\forall i}\{t_i^e\} \le H$ can be satisfied, where t_i^e is the end time of task τ_i. Hence, task duplication and DVFS should be jointly considered during task mapping.



(a) Mapping dependent tasks on single-core ($R = 1$) platform.



(b) Mapping dependent tasks on multi-core ($R = 2$) platform.

Figure 2: Task allocation and scheduling comparisons under different platforms.

3. System Models and Problem Formulation

3.1. System Model

3.1.1. Network Topology Model

We consider a networked system with N wireless nodes $\{\theta_1, \ldots, \theta_N\}$. Network nodes are connected through wireless, having limited communication capability, i.e., one node can communicate with other nodes within its communication range. Furthermore, multi-path data routing is considered, i.e., a pair of nodes can communicate with each other through multiple routing paths, as shown by the red and green arrows in the node graph of Fig. 1. Note that the number of paths between different node pairs may differ. We consider two types of data-routing paths, i.e., time-oriented and energy-oriented, since the problem under study is subject to time and energy constraints. To take multi-path data routing into account, we introduce a node graph $\mathcal{G}_n(\mathcal{V}_n, \mathcal{E}_n)$, where the vertexes $\mathcal{V}_n$ represent the nodes, and the edges $\mathcal{E}_n$ represent the communication cost (i.e., communication time and energy) between the adjacent nodes.

Based on the node graph $\mathcal{G}_n(\mathcal{V}_n, \mathcal{E}_n)$, we obtain an energy matrix $\boldsymbol{e} = [e_{\beta\gamma k\rho}]_{N\times N\times N\times C}$, and a time matrix $\boldsymbol{t} = [t_{\beta\gamma\rho}]_{N\times N\times C}$, where $e_{\beta\gamma k\rho}$ represents the energy consumed at a node θ_k when unit of task data is transmitted from θ_β to θ_γ through the ρ^{th} path, and $t_{\beta\gamma\rho}$ represents the time required to transmit task data from θ_β to θ_γ through the ρ^{th} path. As shown in Fig. 1, the first (path1) and the second (path2) routing paths are assumed to be the time-oriented and energy-oriented paths, respectively. Since the weights of the edges represent the communication costs (e.g., communication time or energy), the aim of time (energy)-oriented routing is to find the shortest path, which can be easily found through the existing methods, such as the Dijkstra algorithm.

3.1.2. Node Model

Each node θ_k is realized with a multicore platform that contains R cores $\{\phi_k^1, \ldots, \phi_k^R\}$. The cores support DVFS technology and each core has L discrete Voltage/Frequency (V/F) levels $\{(v_1, f_1), \ldots, (v_L, f_L)\}$. The relationship between the supply voltage v_l and working frequency f_l is almost linear [7, 6]; when the frequency changes, the voltage also changes. The processor's power, working with a given V/F level (v_l, f_l) [6, 31, 17], is

$$P_l = P_l^s + P_l^d,$$

where $P_l^s = v_l K_1 e^{K_2 v_l} e^{K_3 v_{bs}} + |v_{bs}| I_j$ is the static power, and $P_l^d = C_{eff} v_l^2 f_l$ is the dynamic power. $K_1 - K_3$ are technology-dependent constants. I_j is the approximately constant junction leakage current. v_{bs} is the reverse bias voltage used to reduce leakage power and can be treated as constant. C_{eff} is the average effective switching capacitance [31, 17]. Different cores in one processor communicate with each other through the high-speed data bus. The communication costs among these cores are small enough to be ignored compared to those between different nodes [15]. Therefore, we introduce two small positive values, ϵ and ε, representing the communication time and energy between the cores in one processor. Based on the node graph $\mathcal{G}_n(\mathcal{V}_n, \mathcal{E}_n)$, we can obtain a core graph $\mathcal{G}_c(\mathcal{V}_c, \mathcal{E}_c)$, where the vertexes $\mathcal{V}_c$ represent the cores, and the edges $\mathcal{E}_c$ represent the communication cost between the cores.

3.1.3. Task Model

We consider a real-time application consisting of M dependent and periodic tasks $\mathcal{T} = \{\tau_1, \dots, \tau_M\}$. The tasks are released at time 0 and have the same scheduling horizon H, which can be defined as the least common multiple of task periods [6]. For each task τ_i, W_i is the Worst Case Execution Cycles (WCEC), and D_i is the task deadline. The dependency between the tasks is described by a binary matrix $\boldsymbol{p} = [p_{ij}]_{M\times M}$, where $p_{ij} = 1$ represents that task τ_i precedes task τ_j and τ_j is the closest task of τ_i. For the dependent tasks τ_i and τ_j, i.e., $p_{ij} = 1$, when τ_i is executed, it generates a set of data with size s_{ij} for τ_j. As the example shown in Fig. 1, for the dependent tasks τ_9 and τ_2, we have $p_{92} = 1$, as τ_9 generates a set of data with size s_{92} for τ_2 when finished.

When a task τ_i is executed with a V/F level (v_l, f_l), the reliability of task execution [22] is

$$R_{il} = e^{-\lambda \times 10^{\frac{d(f_{\max}-f_l)}{f_{\max}-f_{\min}}} \times \frac{W_i}{f_l}},$$

where λ is the maximum failure rate, d is a positive constant indicating the sensitivity of failure rate related to frequency scaling, $f_{\max} = \max_{\forall l}\{f_l\}$ and $f_{\min} = \min_{\forall l}\{f_l\}$ are the maximum and minimum frequencies of core, respectively.

Let R_{th} denote the threshold required for the reliability of a task. If the reliability of τ_i is lower than this threshold, to enhance the task reliability, τ_i is duplicated. The common fault detection technologies include consistency checks and response testing [36]. In Fig. 1, τ_{17} and τ_{20} are the replicas of τ_7 and τ_{10}, respectively. Note that by duplicating the tasks, the task dependencies change. Since τ_3 and τ_{10} are dependent, and τ_{20} is a duplicated task of τ_{10}, τ_3 and τ_{20} are also dependent.

3.2. Task Mapping Problem

This work considers static task mapping, optimizing the energy consumption of multi-core nodes under task reliability, real-time, and dependency constraints. During the task mapping process, we determine the following decisions: 1) task frequency assignment, 2) task duplication, 3) task allocation, 4) task sequence, 5) task start time, and 6) path selection. To formulate the task mapping problem, we introduce the following binary and parameters and continuous variables:

1. $y_{il} = 1$, if task τ_i is executed with frequency f_l, otherwise, $y_{il} = 0$;
2. $h_i = 1$, if task τ_i exists, otherwise, $h_i = 0$;
3. $q_{id} = 1$, if task τ_i is allocated to core ϕ_d, otherwise, $q_{id} = 0$;
4. $o_{ij} = 1$, if task τ_i proceeds τ_j, otherwise, $o_{ij} = 0$;
5. $c_{\beta\gamma\rho} = 1$, if task data is routed from θ_β to θ_γ along with the ρ^{th} path, otherwise, $c_{\beta\gamma\rho} = 0$;
6. continuous variable t_i^s denotes the start time of task τ_i.

Table 1
Parameters and variables used in the problem formulation

Parameters	
N	number of nodes
M	number of tasks
P	number of node's cores
L	number of voltage/frequency level
H	scheduling horizon
τ_i	the i^{th} task
θ_k	the k^{th} node
ϕ_d	the d^{th} core
(v_l, f_l)	the l^{th} V/F level
W_i	the worst case execution cycles of τ_i
D_i	deadline of τ_i
p_{ij}	$= 1$, if τ_i and τ_j are dependent, else, $= 0$
s_{ij}	task data size τ_i generates for τ_j
P_l	processor power working on (v_l, f_l)
R_{th}	reliability threshold
R_{il}	reliability of τ_i executed with (v_l, f_l)
$e_{\beta\gamma k\rho}$	energy consumed by θ_k for routing task data from θ_β to θ_γ by the ρ^{th} path
$t_{\beta\gamma\rho}$	communication time between θ_β and θ_γ by the ρ^{th} path
Binary Variables	
q_{id}	$= 1$, if τ_i is assigned to ϕ_d, else, $= 0$
x_{ik}	$= 1$, if τ_i is allocated to θ_k, else, $= 0$
y_{il}	$= 1$, if τ_i is executed with (v_l, f_l), else, $= 0$
o_{ij}	$= 1$, if τ_i precedes τ_j, else, $= 0$
$c_{\beta\gamma\rho}$	$= 1$, if task data is routed from θ_β to θ_γ by the ρ^{th} path, else, $= 0$
h_i	$= 1$, if τ_i exists, else, $= 0$
Continuous Variables	
t_i^s	start time of τ_i

The parameters and the variables mainly used in the problem formulation are summarized in Table 1. For the sake of problem formulation, let $\mathcal{M} = \{1, \dots, M\}$, $\mathcal{N} = \{1, \dots, N\}$, $\mathcal{L} = \{1, \dots, L\}$, and $\mathcal{C} = \{1, \dots, C\}$ denote the sets of tasks, nodes, V/F levels, and data routing paths, respectively. Considering task duplication and multi-core platforms, let $\mathcal{M}' = \{1, \dots, 2M\}$ denote the set of all tasks (including the original and duplicated tasks), and let $\mathcal{N}' = \{1, \dots, NR\} = \{\mathcal{P}_1, \dots, \mathcal{P}_k, \dots, \mathcal{P}_N\}$ denote the set of all cores, where $\mathcal{P}_k$ is the set of cores in the node θ_k. The constraints and the objective function of the task mapping problem are described as follows.

3.2.1. Task Allocation Constraints

Each task is executed on one core, without considering task migration [28], we have

$$\sum\nolimits_{d\in\mathcal{N}'} q_{id} = 1,\ \forall i \in \mathcal{M}'. \tag{1}$$

If task τ_i is executed on core ϕ_d of node θ_k, i.e., τ_i is mapped to core ϕ_d and is assigned to node θ_k at the same time, the variables regarding the task-to-core allocation q_{id} and task-to-node allocation x_{ik} are bounded by

$$\sum\nolimits_{d\in\mathcal{P}_k} q_{id} = x_{ik},\ \forall i \in \mathcal{M}',\ \forall k \in \mathcal{N}. \tag{2}$$

3.2.2. Frequency Selection Constraints

We consider task-level DVFS [6, 7]. Since each task is executed with only one V/F level, the frequency assignment variable y_{il} is bounded by

$$\sum\nolimits_{l\in\mathcal{L}} y_{il} = 1,\ \forall i \in \mathcal{M}'. \tag{3}$$

3.2.3. Path Selection Constraints

Since node θ_β transmits task data to node θ_γ through one data path, we obtain

$$\sum\nolimits_{\rho\in C} c_{\beta\gamma\rho} = 1,\ \forall \beta \neq \gamma \in \mathcal{N}. \tag{4}$$

3.2.4. Task Reliability Constraints

With the frequency assignment variable y_{il}, the reliability of task τ_i without task duplication is $R'_i = \sum_{l\in\mathcal{L}} y_{il} R_{il}$. If $R'_i \geq R_{th}$, there is no need to duplicate task τ_i, and thus, task τ_{i+M} (i.e., the replica of task τ_i) doesn't exist, else (i.e., $R'_i < R_{th}$), task τ_i is duplicated, and task τ_{i+M} exist. To indicate the existence of task τ_i, a binary variable h_i is introduced. Since the original tasks $\{\tau_1, \dots, \tau_M\}$ always exist, while the existences of replicas $\{\tau_{M+1}, \dots, \tau_{2M}\}$ relay on the reliability of original tasks, the relationship between h_i and R'_i can be described as follows: $h_i = 1$ ($\forall i \in \mathcal{M}$) and

$$h_{i+M} = \begin{cases} 0, & R'_i \geq R_{th}, \\ 1, & R'_i < R_{th}, \end{cases} \quad \forall i \in \mathcal{M}. \tag{5}$$

To resolve the comparison problem inside equation (5), we reformulate the task mapping problem by linearly rewriting (5). To achieve that, we introduce the following Lemma.

Lemma 3.1. *Assume that b is a binary variable, and x is a continuous variable bounded by $0 \leq x \leq s$. The comparisons 1) $x \geq s_1 \Rightarrow b = 0$, and 2) $x < s_1 \Rightarrow b = 1$, where $s_1 \leq 1$ is a constant, can be described by a linear function $\frac{x-(s_1-\sigma)}{s} \leq 1 - b \leq \frac{x}{s_1}$, where σ is a positive, small enough value.*

Proof 3.1. *Let $b_1 = \frac{x-(s_1-\sigma)}{s}$ and $b_2 = \frac{x}{s_1}$. We have the following two cases: 1) If $x \geq s_1$, we get $b_2 \geq 1$ and $0 < b_1 < 1$ due to $0 < x - (s_1 - \sigma) < s$. Taking the ranges of b_1 and b_2 into account, as well as $b \in \{0, 1\}$, we obtain $b = 0$; 2) If $x < s_1$, we have $b_2 < 1$ and $b_1 < 0$. Therefore, we get $b = 1$.*

Based on *Lemma* 3.1, we let $\sigma = \min_{\forall i,l} |\{R_{il} - R_{th}\}|$, and then, the comparison (5) can be linearized as follows:

$$\frac{R'_i - (R_{th} - \sigma)}{\max_{\forall i,l}\{R_{il}\}} \leq 1 - h_{i+M} \leq \frac{R'_i}{R_{th}},\ \forall i \in \mathcal{M}. \tag{6}$$

By duplicating the original task τ_i, the reliability of task execution is

$$R_i = 1 - \left(1 - h_i R'_i\right)\left(1 - h_{i+M} R'_{i+M}\right),$$

where $R'_{i+M} = \sum_{l\in\mathcal{L}} y_{(i+M)l} R_{(i+M)l}$ is the reliability of the duplicated task τ_{i+M}, and R_i should satisfy the constraint:

$$R_i \geq R_{th},\ \forall i \in \mathcal{M}. \tag{7}$$

3.2.5. Task Non-overlapping Constraints

Note that some tasks in the task set are independent, e.g., tasks τ_i and τ_j with $p_{ij} = 0$. If τ_i and τ_j are assigned to the same core, we must determine the task execution sequence between τ_i and τ_j since one core cannot execute several tasks simultaneously. To achieve that, we introduce a binary variable o_{ij} leading to the following constraints:

$$t_i^e \leq t_j^s + (2 - q_{id} - q_{jd})H + (1 - o_{ij})H, \\ \forall i \neq j \in \mathcal{M}',\ \forall k \in \mathcal{N}, \tag{8}$$

where $t_i^e = t_i^s + t_i^{comp}$ and $t_i^{comp} = h_i \sum_{l\in\mathcal{L}} y_{il} \frac{W_i}{f_l}$ are the end time and the computation time of τ_i, respectively.

(8) is meaningful only when τ_i and τ_j are allocated to the same core ϕ_d, i.e., $q_{id} = q_{jd} = 1$. If these tasks are allocated to different cores, (8) is always satisfied due to $q_{id} + q_{jd} \leq 1$, which can be omitted. With $q_{id} = q_{jd} = 1$, if $o_{ij} = 1$, (8) is relaxed to $t_i^e \leq t_j^s$, which bounds the start time and the end time of τ_i and τ_j. Since we have $1 - o_{ij} = o_{ji}$ and (8) is for all tasks, (8) can be rewritten as $t_j^e \leq t_i^s + (2 - q_{id} - q_{jd})H + o_{ij}H$. It describes the other case where $o_{ij} = 0$: τ_j should be finished before τ_i starts as $t_j^e \leq t_i^s$. As (8) mainly restricts the start and end time of tasks assigned to the same core, the communication time between τ_i and τ_j can be omitted. They are very small compared to the communication time between the nodes.

3.2.6. Task Dependency Constraints

Based on the task dependency matrix $\boldsymbol{p}$, the number of tasks that precede task τ_i is known, i.e., $\sum_{j\in\mathcal{M}'} p_{ji}$. To execute a task τ_i, we should collect all the data generated from its predecessors. Note that the existence of τ_i is determined by the variable h_i, and τ_i will not generate data for or receive data from other tasks if τ_i does not exist. To avoid communication collision, one node receives the data from other nodes in sequence, since a node cannot receive data from multiple nodes simultaneously. Hence, the time spent for receiving data required by the execution of task τ_i is

$$t_i^{comm} = \sum_{j\in\mathcal{M}'} \sum_{\beta\in\mathcal{N}} \sum_{\gamma\in\mathcal{N}} \sum_{\rho\in C} p_{ji} h_i h_j x_{j\beta} x_{i\gamma} c_{\beta\gamma\rho} t_{\beta\gamma\rho}.$$

For the dependent tasks, e.g., tasks τ_i and τ_j with $p_{ij} = 1$, no matter whether they are assigned to the same core or different cores, the execution sequence between them is fixed. Thus, we have the following constraints:

$$t_i^e + t_j^{comm} \le t_j^s,\ \forall i \ne j \in \mathcal{M}', \tag{9}$$

where (9) implies that if τ_i is the predecessor of τ_j, the start time of τ_j should be later than the end time of τ_i plus the date receiving time of τ_j.

Remark 3.1. *This paper does not explicitly consider priorities among the tasks. The proposed approach can support task priorities, which can be used to define an ordering in the way the tasks are scheduled and executed. However, task priorities also have a higher impact when preemptive execution is considered, which is left as future work.*

3.2.7. Real-time Constraints

Since each real-time task τ_i should be finished within its relative deadline D_i, and the end time of τ_i should small than the scheduling horizon H, we have

$$t_i^{comp} \le D_i,\ \forall i \in \mathcal{M}', \tag{10}$$
$$t_i^e \le H,\ \forall i \in \mathcal{M}'. \tag{11}$$

3.2.8. Objective Function and Primary Problem

If tasks τ_i and τ_j ($p_{ij} = 1$) are dependent, and they are allocated to different nodes, e.g., θ_β and θ_γ where $\beta \ne \gamma$, the energy consumed for node θ_k to transmit the task data from θ_β to θ_γ through the ρ^{th} path is $p_{ij}s_{ij}x_{i\beta}x_{j\gamma}c_{\beta\gamma\rho}e_{\beta\gamma k\rho}$. Considering the task duplication decision h_i, the communication energy of θ_k is

$$E_k^{comm} = \sum_{i\in\mathcal{M}'}\sum_{j\in\mathcal{M}'}\sum_{\beta\in\mathcal{N}}\sum_{\gamma\in\mathcal{N}}\sum_{\rho\in\mathcal{C}} p_{ij}s_{ij}h_ih_jx_{i\beta}x_{j\gamma}c_{\beta\gamma\rho}e_{\beta\gamma k\rho}.$$

On the other hand, if task τ_i is assigned to core ϕ_d and executed by the V/F level (v_l, f_l), i.e., $q_{id} = y_{il} = 1$, the time and the energy required for the task execution are $q_{id}y_{il}\frac{W_i}{f_l}$ and $q_{id}y_{il}\frac{W_i}{f_l}P_l$, respectively. Hence, taking the decisions q_{id}, y_{il}, and h_i into account, the computation energy of θ_k is

$$E_k^{comp} = \sum\nolimits_{i\in\mathcal{M}'}\sum\nolimits_{d\in\mathcal{P}_k}\left(q_{id}h_i\sum\nolimits_{l\in\mathcal{L}} y_{il}\frac{W_i}{f_l}P_l\right).$$

Let E_k^{ini} denote the initial energy of θ_k. To increase the lifetime and connectivity of the system, task mapping aims to balance the energy consumption of the nodes under the above constraints. Therefore, the Primary Problem (PP) can be formulated as follows:

$$\textbf{PP}: \Phi = \min_{\substack{q,x,y,\\ o,c,h,t^s}} \max_{\forall k}\left\{\left(E_k^{comm} + E_k^{comp}\right)/E_k^{ini}\right\} \tag{12}$$

$$\text{s.t.}\begin{cases}(1),(2),(3),(4),(6),(7),(8),(9),(10),(11)\\ q_{id}, x_{ik}, y_{il}, o_{ij}, c_{\beta\gamma\rho}, h_i \in \{0,1\},\ 0 \le t_i^s \le H.\end{cases}$$

Since h_iy_{il}, $h_ih_jx_{i\beta}x_{j\gamma}c_{\beta\gamma\rho}$, and $q_{id}h_iy_{il}$ are the nonlinear items, PP is a MINLP, which is difficult to solve directly.

Theorem 3.1. *The energy-aware task mapping problem based on DVFS, task duplication, and data routing is $\mathcal{NP}$-hard.*

Proof 3.2. *Please refer to [4] for the details.*

3.2.9. Problem Linearization

A linearization method based on variable replacement is used to find the optimal solution to PP and deal with the nonlinear items. Since h_i, y_{il}, x_{ik}, and $c_{\beta\gamma\rho}$ are the binary variables, according to the characteristics of the nonlinear items, we introduce the following lemma.

Lemma 3.2. *Let x, y, and z denote the binary variables. The nonlinear item $z = xy$ can be linearized as follows: $z - x \le 0$, $z - y \le 0$ and $x + y - z \le 1$.*

Proof 3.3. *According to the constraints $z - x \le 0$, $z - y \le 0$ and $x + y - z \le 1$, we obtain the following four cases: 1) $x = 0, y = 0 \Rightarrow z = 0$; 2) $x = 0, y = 1 \Rightarrow z = 0$; 3) $x = 1, y = 0 \Rightarrow z = 0$; 4) $x = 1, y = 1 \Rightarrow z = 1$, which is equal to $z = xy$.*

Based on *Lemma* 3.2, we introduce an auxiliary variable a_{idl} to replace the nonlinear term $q_{id}y_{il}$, and add the following linear constraints into problem (12):

$$a_{idl} + q_{id} \le 0,\ a_{idl} + y_{il} \le 0,\ q_{id} + y_{il} - a_{idl} \le 1,\\ \forall i \in \mathcal{M}',\ \forall d \in \mathcal{R},\ \forall l \in \mathcal{L}. \tag{13}$$

On this basis, we linearize $q_{id}h_iy_{il}$ and set $w_{idl} = h_ia_{idl}$ since $a_{idl} = h_ia_{idl}$:

$$w_{idl} + h_i \le 0,\ w_{idl} + a_{idl} \le 0,\ h_i + a_{idl} - w_{idl} \le 1,\\ \forall i \in \mathcal{M}',\ \forall d \in \mathcal{R},\ \forall l \in \mathcal{L}. \tag{14}$$

Similarly, to deal with nonlinear item $h_ih_jx_{i\beta}x_{j\gamma}c_{\beta\gamma\rho}$, we set $g_{i\beta j\gamma} = x_{i\beta}x_{j\gamma}$, $s_{i\beta j\gamma\rho} = g_{i\beta j\gamma}c_{\beta\gamma\rho}$, $d_{ij} = h_ih_j$, and $e_{i\beta j\gamma\rho} = s_{i\beta j\gamma\rho}d_{ij}$, and add the following linear constraints into PP:

$$g_{i\beta j\gamma} + x_{i\beta} \le 0,\ g_{i\beta j\gamma} + x_{j\gamma} \le 0,\ x_{i\beta} + x_{j\gamma} - g_{i\beta j\gamma} \le 1,\\ \forall i \ne j \in \mathcal{M}',\ \forall \beta \ne \gamma \in \mathcal{N}, \tag{15}$$

$$s_{i\beta j\gamma\rho} + g_{i\beta j\gamma} \le 0,\ s_{i\beta j\gamma\rho} + c_{\beta\gamma\rho} \le 0,\ c_{\beta\gamma\rho} + g_{i\beta j\gamma} - s_{i\beta j\gamma\rho} \le 1,\\ \forall i \ne j \in \mathcal{M}',\ \forall \beta \ne \gamma \in \mathcal{N},\ \forall \rho \in \mathcal{C}, \tag{16}$$

$$d_{ij} + h_i \le 0,\ d_{ij} + h_j \le 0,\ h_i + h_j - d_{ij} \le 1,\\ \forall i \ne j \in \mathcal{M}', \tag{17}$$

$$e_{i\beta j\gamma\rho} + d_{ij} \le 0,\ e_{i\beta j\gamma\rho} + s_{i\beta j\gamma\rho} \le 0,\ d_{ij} + s_{i\beta j\gamma\rho} - e_{i\beta j\gamma\rho} \le 1,\\ \forall i \ne j \in \mathcal{M}',\ \forall \beta \ne \gamma \in \mathcal{N},\ \forall \rho \in \mathcal{C}. \tag{18}$$

By using the auxiliary variables a_{idl}, w_{idl}, $g_{i\beta j\gamma}$, $s_{i\beta j\gamma\rho}$, d_{ij}, and $e_{i\beta j\gamma\rho}$ and the additional constraints (13)-(18), PP (12) can be transformed into the following MILP problem:

$$\mathbf{PP1}: \min_{\substack{q,x,y,o,c,h,t^s,\\ a,w,g,s,d,e}} \max_{\forall k} \left\{(E_k^{comm} + E_k^{comp})/E_k^{ini}\right\} \quad (19)$$

$$\text{s.t.} \begin{cases} (1)-(4), (6), (7), (8), (9)-(11), (13)-(18) \\ q_{id}, x_{ik}, y_{il}, o_{ij}, c_{\beta\gamma\rho}, h_i, a_{idl}, w_{idl}, g_{i\beta j\gamma}, s_{i\beta j\gamma\rho}, \\ d_{ij}, e_{i\beta j\gamma\rho} \in \{0,1\},\ 0 \le t_i^s \le H. \end{cases}$$

Remark 3.2. Lemma *3.2 shows that the linearization does not change the feasible region of the problem. Therefore, solving the problems with and without linearization is equivalent.*

3.2.10. Discussion

The additional overhead associated with applying DVFS occurs at the initial and final moments of V/F level transitions. The time and energy overheads can be modeled as

$$t^{switch}(f_s, f_e) = t_{trans} \times \frac{(f_e - f_s)}{f_e},$$
$$E^{switch}(f_s, f_e) = P^{switch} \times t^{switch}(f_s, f_e),$$

where t_{trans} is the time cost of V/F level transition, f_s and f_e are the frequency levels at the start instant and the end instant of the V/F level transition, and P^{switch} is the power before the V/F level transition. Based on the parameters of multiple core nodes, we can obtain the computational overhead t_i^{switch} and E_i^{switch} for task τ_i when computed on the cores, as well as the communication overhead t_{ij}^{switch} and E_{ij}^{switch} for data transmission over the edge e_{ij}. When considering the overhead introduced by DVFS in problem (12), these additional overheads should be incorporated into the constraints (8), (9), (10), and (11).

The proposed task mapping method can be extended to heterogeneous multicore platforms by modifying the constraint (3), where the frequency assignment variable y_{il} is replaced by y_{ikl}, and $y_{ikl} = 1$ represents task τ_i is allocated to the core of θ_k and executed by the l_{th} V/F level. With DVFS, the execution time of task τ_i on the node θ_k is updated to

$$t_i^{comp} = \sum_{k\in N} \sum_{l\in L} y_{ikl} \frac{W_i}{f_{kl}},$$

and the reliability of task τ_i is updated to

$$R_i = \sum_{k\in N} \sum_{l\in L} y_{ikl} e^{-\lambda \times 10^{\frac{d(f_{\max} - f_{kl})}{f_{\max} - f_{\min}}} \times \frac{W_i}{f_{kl}}}.$$

In some applications, the task-to-node allocation is restricted, e.g., the sensing tasks are allocated to the sensor nodes, the control tasks are allocated to the actuator nodes, while the data processing tasks can be allocated to the nodes with communication and computation capabilities. In this context, additional constraints should be added to the task mapping problem: $\sum_{k\in\mathcal{N}^s} x_{ik} = 1, \forall i \in \mathcal{M}^s$, when $\mathcal{N}^s$ and $\mathcal{M}^s$ are the sets of nodes and tasks that have the links, respectively.

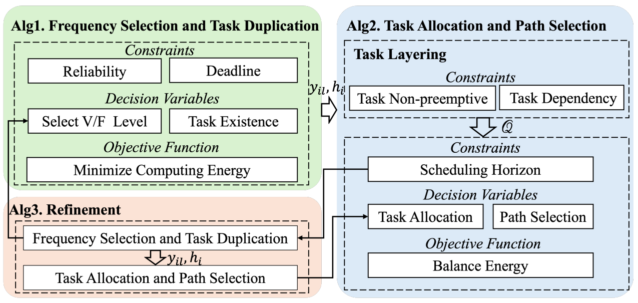


Figure 3: The structure of the proposed heuristic method.

4. Heuristic Algorithm for Task Mapping

Since the transformed problem PP1 (19) is a MILP, we can use existing optimal methods, e.g., Branch and Bound (B&B) and Benders Decomposition (BD) [17], and optimization solvers, e.g., Gurobi and Cplex [6], to solve it. However, finding the optimal solution is still time-consuming, especially when the size of the problem is large. To enhance the scalability of the proposed method, we design a novel heuristic approach for solving the problem (12). As shown in the previous section, problems (12) and (19) are equivalent, with problem (12) having fewer variables and constraints. Thus, we target the original problem (12) rather than the linearized problem (19). The proposed heuristic is based on the problem decomposition principle. For an optimization problem, the computational complexity is related to the number of variables and constraints. Therefore, dividing the original problem into subproblems solved sequentially is more efficient than solving the original problem. We observe that adjusting the V/F level does not affect the decisions in task allocation as the node cores are homogeneous (i.e., they have the same V/F levels). Therefore, the proposed heuristic includes two main steps, i.e., the frequency selection and task replication, and the task allocation and path selection, combined with a refinement process. Fig. 3 shows the structure of the proposed heuristic method.

4.1. Problem Decomposition and Refinement Process

4.1.1. Frequency Assignment and Task Duplication (FATD)

In this step, we decide the frequency assignment y_{il} and task replication h_i. Note that the values of y_{il} and h_i will influence the reliability constraint (7) and the relative deadline constraint (10). In addition, y_{il} is restricted by the frequency selection constraint (3), h_i is determined by y_{il} through (6). The goal of the task mapping problem (19) is to balance the energy consumption of the networked system. Therefore, in this step, we reduce the task execution energy $E_i^{comp} = \sum_{l\in\mathcal{L}} y_{il} \frac{W_i}{f_l} P_l$ under the constraints (3), (6), (7), (10). Although y_{il} and h_i influence also the real-time constraint (11), the task-to-node allocation x_{ik}, task-to-core allocation q_{id}, and path selection $c_{\beta\gamma\rho}$ are unknown at this step. This step does not consider the constraint (11) and the communication energy. Hence, the Frequency Assignment and Task Duplication (FATD) problem can be formulated as:

$$\textbf{FATD}: \Phi_1 = \min_{\boldsymbol{y},\boldsymbol{h}} \sum\nolimits_{i\in\mathcal{M}'} h_i \sum\nolimits_{l\in\mathcal{L}} y_{il}\frac{W_i}{f_l}P_l \qquad (20)$$

$$\text{s.t.}: \begin{cases} (3),(6),(7),(10) \\ y_{il}, h_i \in \{0,1\}. \end{cases}$$

Note that the binary variables y_{il} and h_i are coupled with each other nonlinearly in (20), and thus, FATD is an INLP problem. To solve it, we determine the frequency selection y_{il} and the task duplication h_i in sequence, as h_i is influenced by y_{il}, according to (6). Specifically, for each original task τ_i ($\forall i \in \mathcal{M}$), when the V/F level (v_l, f_l) is assigned to τ_i, the value of y_{il} can be determined by minimizing the computation energy E_i^{comp} under the relative deadline constraint (10). Then, based on the reliability of the original task $R_i' = \sum_{l\in\mathcal{L}} y_{il}R_{il}$, the existence of the duplicated task, i.e., h_{i+M}, is determined according to (5) or (6). The frequency selection for the duplication can be determined using a method similar to the original tasks. Therefore, the variables and the constraints in (20) are considered separately. Although this method has a simpler structure with low computation complexity, it may significantly impact performance.

To avoid that, we design a greedy-based heuristic method that simultaneously considers y_{il} and h_i, depicted in Algorithm 1. For presentation reasons, a frequency assignment index $B[i]$ is introduced for each task τ_i, where $B[i] = l$ represents that the l^{th} V/F level (v_l, f_l) is used to execute task τ_i, and the value of $B[i]$ is initialized as -1 (Line 1). The algorithm applies a set of steps to determine a set of configurations for each original task τ_i that optimize y_{il} and h_i under multiple constraints:

1. We use the sequence $\{(v_1, f_1), \ldots, (v_L, f_L)\}$ for assigning a V/F level (v_l, f_l) to an original task τ_i ($\forall i \in \mathcal{M}$). When τ_i is executed with (v_l, f_l), the task execution time is $\frac{W_i}{f_l}$. If this time does not satisfy the deadline constraint (10) (Lines 5–6), the V/F selection $B[i] = l$ is excluded, and thus, $y_{il} = 0$.
2. When the task execution time with the V/F level (v_l, f_l) satisfies the task relative deadline constraint, of τ_i, i.e., $\frac{W_i}{f_l} \le D_i$, it implies that with higher V/F levels, i.e., $\{(v_{l+1}, f_{l+1}), \ldots, (v_L, f_L)\}$, the relative deadline constraint of τ_i is also satisfied. This is because the V/F levels in the set $\{(v_1, f_1), \ldots, (v_L, f_L)\}$ are sorted in increasing order.
3. Based on a given V/F level (v_j, f_j) $(l \le j \le L)$, the reliability of original task τ_i is known. By comparing R_i' with the reliability threshold R_{th} (Line 9), we determine whether τ_i needs duplication or not, and thus, the value of h_{i+M}. If the reliability of τ_i satisfies the constraint (7), no duplication is required, i.e., $h_{i+M} = 0$. Then, we can calculate the computation energy $\frac{W_i}{f_j}P_j$ of τ_i.
4. If $R_i' < R_{th}$, task τ_i needs duplication (Line 15), i.e., $h_{i+M} = 1$ as τ_{i+M} exists. Note that the relative deadline constraint (10) also restricts the duplicated tasks. According to $1-\left(1-R_i'\right)\left(1-R_{i+M}'\right) \ge R_{th}$, $R_{i+M}' = \sum_{l\in\mathcal{L}} y_{(i+M)l}R_{(i+M)l}$, and $t_i^{comp} \le D_i$, we can calculate the minimum V/F level for the duplicated task τ_{i+M} (Line 17) to make sure that task τ_i with duplication satisfies the reliability and time constraints (7) and (10), e.g., $y_{(i+M)j'}$ (Line 18). Then, we calculate the computation energy for original and duplicated tasks under the given V/F levels (v_j, f_j) and $(v_{j'}, f_{j'})$, i.e., $\frac{W_i}{f_j}P_j$ and $\frac{W_i}{f_{j'}}P_{j'}$.
5. Besides (v_j, f_j), that meets the task relative deadline constraint, we can use a higher V/F level (v_{j+1}, f_{j+1}) to execute task τ_i and calculate the corresponding computation energy $\frac{W_i}{f_{(j+1)}}P_{(j+1)}$ and $\frac{W_i}{f_{(j+1)'}}P_{(j+1)'}$ by a similar method. If τ_i does not need to be duplicated under the given V/F level (v_{j+1}, f_{j+1}), we have $\frac{W_i}{f_{(j+1)'}}P_{(j+1)'} = 0$.
6. Assume that V/F levels $\{(v_l, f_l), \ldots, (v_L, f_L)\}$ satisfy the reliability and time constraints. By using them, we get a set of configurations for the computation energy

Algorithm 1: Frequency Selection and Task Duplication

```
Input: $W_i$, $D_i$, $(v_l, f_l)$, and $R_{th}$
Output: Frequency assignment $y_{il}$, and task duplication $h_i$
1  Initialize $B[i] = -1$ ;
2  for $\forall i \in \mathcal{M}$ do
3      $E_{\min}^{comp} = \infty$ ;
4      for $\forall l \in \mathcal{L}$ do
5          if $\frac{W_i}{f_l} > D_i$ then
6              Continue;
7          else
8              $R_i' = \sum_{l\in\mathcal{L}} y_{il}R_{il}$ ;
9              if $R_i' > R_{th}$ then
10                 $h_{i+M} = 0$ ;
11                 $E_i^{comp} = \frac{W_i}{f_l}P_l$ ;
12                 if $E_i^{comp} < E_{\min}^{comp}$ then
13                     $E_{\min}^{comp} = E_i^{comp}$ ;
14                     $B[i] = l$ ;
15             else
16                 $h_{i+M} = 1$ ;
17                 Calculate $R_{i+M}'$ by
                     $1-(1-R_i')(1-R_{i+M}') \ge R_{th}$ ;
18                 Calculate minimum frequency $y_{(i+M)j'}$
                     by $R_{i+M}' = \sum_{l\in\mathcal{L}} y_{(i+M)l}R_{(i+M)l}$ and
                     $t_i^{comp} \le D_i$ ;
19                 $E_i^{comp} = \frac{W_i}{f_l}P_l + \frac{W_i}{f_{l'}}P_{l'}$ ;
20                 if $E_i^{comp} < E_{\min}^{comp}$ then
21                     $E_{\min}^{comp} = E_i^{comp}$ ;
22                     $B[i] = l$ ;
23                     $B[i+M] = l'$ ;
```

Algorithm 2: Task Allocation and Path Selection

```
Input: Frequency assignment $y_{il}$, task duplication $h_i$, and
       task execution sequence $\mathcal{Q}$
Output: Task allocation $q_{id}$ and $x_{ik}$, path selection $c_{\beta\gamma\rho}$,
        task order $o_{ij}$, task start time $t_i^s$
Initialize $O[i] = C[j][i] = ST[i] = -1$;
for $\forall i \in \mathcal{M}'$ and $h_i = 1$ do
    $E^{total}_{\min,\max} = \infty$;
    for $\forall d \in \mathcal{N}'$ do
        for $\forall \tau_j \in prec(\tau_i)$ do
            for $\forall \rho \in C$ do
                $t_i^s = \max_{\forall \tau_j \in pre(\tau_i)}\{t_j^e + t^{comm}_{ji\rho}, t^e_{j'}\}$ while
                  $O[j'] = d$;
                $t_i^e = t_i^s + t_i^{comp}$;
                if $t_i^e > H$ then
                    continue;
                $E^{total}_{i,\max} = \max_{\forall k \in \mathcal{N}}\{E_k^{comp} + E_k^{comm}\}$;
                if $E^{total}_{i,\max} < E^{total}_{\min,\max}$ then
                    $E^{total}_{\min,\max} = E^{total}_{i,\max}$;
                    $O[i] = d$;
                    $C[j][i] = \rho$;
```

of task τ_i, $\left\{\frac{W_i}{f_l}P_l + \frac{W_i}{f_{l'}}P_{l'}, \dots, \frac{W_i}{f_L}P_L + \frac{W_i}{f_{L'}}P_{L'}\right\}$, and find the minimum one (Line 21). If the V/F levels with the minimum computation energy are (v_l, f_l) and $(v_{l'}, f_{l'})$, we select the V/F level $B[i] = l$ for the original task τ_i and the V/F level $B[i+M] = l'$ for the duplicated task τ_{i+M} (Lines 22–23). Then, we determine y_{il} and h_i through $B[i]$ $(1 \le i \le 2M)$.

4.1.2. Task Allocation and Path Selection (TAPS)

When frequency assignment y_{il} and task duplication h_i decisions are determined, the number of total tasks (including original and duplicated tasks) and the computation costs (energy and time) are known. Note that the relative real-time requirement (10) has been met, and the task allocation decision does not affect task execution time. Therefore, this step aims to balance the energy consumption of the nodes by task-to-core allocation q_{id}, task-to-node allocation x_{ik}, and path selection $c_{\beta\gamma\rho}$ under task non-overlapping constraint (8), task dependency constraint (9), and task deadline constraint (11). The Task Allocation and Path Selection (TAPS) problem can be formulated as follows:

$$\textbf{TAPS}: \Phi_2 = \min_{\boldsymbol{q},\boldsymbol{x},\boldsymbol{o},\boldsymbol{c},\boldsymbol{t}^s} \max_{\forall k}\{E_k/E_k^{ini}\} \tag{21}$$

$$\text{s.t.} \begin{cases} (1),(2),(4),(8),(9),(11) \\ q_{id}, x_{ik}, o_{ij}, c_{\beta\gamma\rho} \in \{0,1\},\ 0 \le t_i^s \le H. \end{cases}$$

Since the binary variable q_{id}, x_{ik}, $c_{\beta\gamma\rho}$, o_{ij} and the continuous variable t_i^s are coupled with each other nonlinearly in (9) and (11), TAPS problem is a MINLP. To effectively solve this problem, we propose a task layer classification method to determine the sequence of tasks during task allocation, satisfying task non-overlapping constraint (8) and task dependency constraint (9). The proposed method is summarized as follows.

1. Based on h_i, we derive a new task set $\mathcal{T} = \{\tau_i | h_i = 1, 1 \le i \le 2M\}$, where the tasks in $\mathcal{T}$ exist, and we need to deploy these tasks to the cores of the nodes. Let M^h denote the number of tasks in the set $\mathcal{T}$. According to $\mathcal{T}$, we obtain a new matrix $\boldsymbol{p}^h = [p^h_{ij}]_{M^h \times M^h}$ to model the dependency of tasks in the set $\mathcal{T}$. To perform task stratification, we introduce the in-degree and out-degree for task τ_i and denote them as $\mathcal{D}_i^{In}$ and $\mathcal{D}_i^{Out}$, respectively. For a task τ_i, its $\mathcal{D}_i^{In} = \sum_{j=1}^{M^h} p^h_{ji}$, while its $\mathcal{D}_i^{Out} = \sum_{j=1}^{M^h} p^h_{ij}$. Note that for an entry task, $\mathcal{D}_i^{In} = 0$, as it has no predecessors; while for an exit task, $\mathcal{D}_i^{Out} = 0$, as it has no successors.
2. The task layer classification starts from the tasks with $\mathcal{D}_i^{In} = 0$ and ends at the tasks with $\mathcal{D}_i^{Out} = 0$. Let $L_i = \max_{\forall p^h_{ji}=1}\{L_j\} + 1$ denote the layer index of task τ_i, i.e., the layer index of τ_i equals the maximum level of all its predecessors plus one. For instance, in the example of Fig. 1, we have $L_6 = L_3 = 1$, $L_1 = L_9 = L_{10} = L_{20} = 2$, $L_2 = 3$, $L_4 = L_8 = 4$, and $L_5 = L_7 = L_{17} = 5$. Tasks with adjacent layers are dependent (e.g., τ_6 and τ_1), while tasks in the same layers are independent (e.g., τ_1 and τ_9). Based on the task layer index L_i, we generate a task execution sequence $\mathcal{Q} = \{L_1 = 1, \dots, L_i = m, L_j = m+1, \dots\}$. For the tasks in the same layer, we sort them in ascending order based on their execution cycles W_i. By following $\mathcal{Q}$ during task allocation, the task non-overlapping constraint (8) and the task dependency constraint (9) can be met simultaneously.

On this basis, we design a greedy-based heuristic algorithm to solve problem (21), which considers q_{id}, x_{ik}, and $c_{\beta\gamma\rho}$ simultaneously. The implementation details are summarized in Algorithm 2. For the sake of presentation, we introduce the task allocation index $O[i]$, path selection index $C[j][i]$, and task start time index $ST[i]$, where $B[i] = d$ represents that task τ_i is allocated on core ϕ_d, $C[j][i] = \rho$ represents that τ_j transmits data to τ_i through the ρ^{th} path, and $ST[i] = t$ represents that τ_i starts its execute at time t. They are initialized to -1 (Line 1).

1. We follow the sequence $\mathcal{Q}$ to assign task $\tau_i \in \mathcal{T}$ to each core ϕ_d $(\forall d \in \mathcal{N}')$. Note that the range of d is $\mathcal{N}'$. We consider q_{id} and x_{ik} at the same time. In addition, the cores of the nodes are homogeneous, i.e., they have the same V/F levels. According to the frequency assignment decision y_{il} from Algorithm 1, we can calculate the computation time $t_i^{comp} = \sum_{l\in\mathcal{L}} y_{il}\frac{W_i}{f_l}$, and computation energy $E_i^{comp} = \sum_{l\in\mathcal{L}} y_{il}\frac{W_i}{f_l}P_l$ of ϕ_d, when τ_i is executed on ϕ_d (Lines 2-4).
2. Let $pre(\tau_i)$ denote the predecessor set of task τ_i. Note that the tasks in $\mathcal{Q}$ are sorted by their in-degree $\mathcal{D}_i^{In}$ and out-degree $\mathcal{D}_i^{Out}$. During the allocation process of τ_i, the allocation of its predecessor, e.g., $\tau_j \in pre(\tau_i)$, is known. When the task

τ_i is mapped on the core ϕ_d, and the path c_ρ is selected to transmit task data between τ_i and its predecessor τ_j, we can calculate the communication time $t_{ji\rho}^{comm} = \sum_{\beta,\gamma\in\mathcal{N}} p_{ji}x_{j\beta}x_{i\gamma}c_{\beta\gamma\rho}t_{\beta\gamma\rho}$ and energy $E_{ji\rho}^{comm} = \sum_{\beta,\gamma\in\mathcal{N}} p_{ji}s_{ji}x_{j\beta}x_{i\gamma}c_{\beta\gamma\rho}e_{\beta\gamma\gamma\rho}$ (Lines 5-6).

3. Based on the computation time t_i^{comp}, the communication time $t_{ji\rho}^{comm}$, and the allocation of task τ_i, we can obtain task start time t_i^s. The value of t_i^s is influenced by a) the end time of predecessor τ_j and the communication time $t_{ji\rho}^{comm}$ between τ_j and τ_i, i.e., t_j^e and $t_{ji\rho}^{comm}$, and b) the end time of the earlier task on the same core ϕ_d, e.g., $\tau_{j'}$ where $O[j'] = d$. Note that τ_i has multiple predecessors, and $\tau_{j'}$ is the last task executed on the same core ϕ_d before τ_i. Restricted by the task dependency, the start time of τ_i can be set to $t_i^s = \max_{\forall\tau_j\in pre(\tau_i)}\left\{t_j^e + t_{ji\rho}^{comm}, t_{j'}^e\right\}$, where $t_{j'}^e$ is the end time of task $\tau_{j'}$ (Line 7). Based on the updated start time t_i^s and computation time t_i^{comp} from Algorithm 1, we can calculate task end time t_i^e, i.e., $t_i^e = t_i^s + t_i^{comp}$ (Line 8).
4. If the end time t_i^e of τ_i does not satisfy the real-time constraint (11) (Lines 9-10), the decisions of task allocation $O[i] = d$ and path selection $C[j][i] = \rho$ will be excluded, and thus, we have $q_{id} = 0$ and $c_{O[j]O[i]\rho} = 0$.
5. If t_i^e satisfies the real-time constraint, i.e., $t_i^e \le H$, we can get the total computation and communication energy for τ_i under the given core ϕ_d and data path between τ_j and τ_i, i.e., $E_i^{comp} + E_{ji\rho}^{comm}$. Since we have N nodes and each node has R cores, by allocating task τ_i to cores $\{\phi_1, \dots, \phi_{NR}\}$, we can get the computation and communication energy on the nodes $\{\theta_1, \dots, \theta_N\}$, i.e., $\left\{E_1^{comp} + E_1^{comm}, \dots, E_N^{comp} + E_N^{comm}\right\}$. We find the maximum one by comparing these values and denote it as $E_{i,\max}^{total}$ (Line 11).
6. By applying the above method for each task τ_i ($1 \le i \le M^h$) in the set $\mathcal{T}$, we obtain a set of task allocation and path selection configurations $\left\{E_{1,\max}^{total}, \dots, E_{M^h,\max}^{total}\right\}$, based on the total energy consumption. To balance the energy consumption of the nodes, we can find the minimum one and denote it as $E_{\min,\max}^{total}$ (Line 13). Through $E_{\min,\max}^{total}$, we can determine task allocation $O[i] = d$, path selection $C[j][i] = \rho$ between tasks τ_j and τ_i, and task start time $ST[i]$ (Lines 14-15). On this basis, we can get the values of optimization variables q_{id}, x_{ik}, o_{ij}, $c_{\beta\gamma\rho}$, and t_i^s.

4.1.3. *Refinement Process*

When solving the problem (21), it may be infeasible because the real-time constraint (11) is hard to satisfy, while other constraints (8) and (9) can be easily satisfied according to the proposed task classification method. In this section, we design a refinement method to improve the task schedulability of problem (21). Note that the adjustment of V/F assignment y_{il} changes the task execution time t_i^{comp} and end time t_i^e, and it also influences the task duplication h_i. Therefore, the basic idea is to adjust y_{il} and h_i through constraint (11). Therefore, a feedback mechanism is introduced from *Step 2* (TAPS) to *Step 1* (FATD). Since the refinement increases the V/F assignment y_{il} in *Step 1*, the reliability and time constraints (7) and (10) in (20) are not violated. The details (Algorithm 3) are summarized below:

1. We find the task τ_i with $t_i^e > H$ in Algorithm 2 and obtain the corresponding V/F level y_{il} from Algorithm 1. On this basis, we get the minimum V/F level (v_{l+a}, f_{l+a}) for τ_i that meets the real-time constraint $t_i^e \le H$, where $f_{l+a} \ge \frac{W_i}{H-\frac{W_i}{f_l}}$. Since the V/F levels are sorted by increasing order, applying the higher V/F levels $\{(v_{l+a}, f_{l+a}), \dots, (v_L, f_L)\}$ to execute τ_i also satisfies the time constraints (10) and (11)(Lines 2-3).
2. Based on the given V/F level (v_{l+a}, f_{l+a}), Algorithm 1 is run to calculate the reliability R_i' and the duplication h_{i+M} of task τ_i (Lines 4). When the above method is applied to each V/F level in the set $\{(v_{l+a}, f_{l+a}), \dots, (v_L, f_L)\}$, we can obtain the corresponding computation energy $\left\{\frac{W_i}{f_{l+a}}P_{l+a} + h_{i+M}\frac{W_i}{f_{l+a'}}P_{l+a'}, \dots, \frac{W_i}{f_L}P_L + h_{i+M}\frac{W_i}{f_{L'}}P_{L'}\right\}$, and select the minimum one as $E_{i,\min}^{comp}$ (Lines 6-10). On this basis, we can update y_{il} and h_i at the same time to meet the time constraint (11) and the reliability constraint (7).
3. Note that multiple tasks may violate the time constraint (11) in Algorithm 2, and they may have dependencies. To reduce problem complexity, *Step 1* and *Step 2* are performed during the task allocation and scheduling process of Algorithm 2. For instance, in Lines 8-10, if the end time t_i^e of τ_i exceeds the time threshold H, we adjust the V/F level y_{il} of τ_i. Then, based on the updated y_{il} and h_i, we calculate the corresponding q_{id}, x_{ik}, $c_{\beta\gamma\rho}$, o_{ij}, and t_i^s. Since we follow the task sequence Q to adjust the V/F levels of tasks, the task non-overlapping constraint (8) and the task dependency constraint (9) can be satisfied.

4.2. Complexity Analysis

The proposed heuristic divides the PP (12) into two subproblems FATD (20) and TAPS (21) and solves them in sequence through Algorithm 1 and Algorithm 2. The refinement method (Algorithm 3) is added during the execution process of Algorithm 2 to improve task schedulability. Therefore, a feedback mechanism is introduced between Algorithm 1 and Algorithm 2.

1. The time complexity of the Algorithm 1 is given by the number of variables [13] in problem (20), i.e., $\boldsymbol{y} = [y_{il}]_{M\times L}$ and $\boldsymbol{h} = [h_i]_{M\times 1}$. For each original task τ_i ($\forall i \in \mathcal{M}$), we calculate task computation time t_i^{comp} and energy E_i^{comp} and reliability R_i' through the V/F levels $\{(v_1, f_1), \dots, (v_L, f_L)\}$, and then determine the duplication h_{i+M} of τ_i based on threshold R_{th}. A similar method is used for the duplicated task τ_{i+M}.

Algorithm 3: Refinement Process

```
Input: $y_{il}$ and $h_i$ from Algorithm 1, $\tau_i$ with $t_i^e > H$ in
       Algorithm 2
Output: $y_{il}$, $h_i$, $q_{id}$, $x_{ik}$, $c_{\beta\gamma\rho}$, $o_{ij}$, and $t_i^s$
1  for $\forall i \in \mathcal{M}'$ & $h_i = 1$ do
2      if $t_i^e > H$ then
3          Find V/F level $\{(v_{l+a}, f_{l+a})\}$ by $t_i^e \geq H$ ;
4          Run Algorithm 1 with
             $\{(v_{l+a}, f_{l+a}), \dots, (v_L, f_L)\}$;
5          $E_{i,\min}^{comp} = \infty$;
6          for $\forall j \in \{l+a, \dots, L\}$ do
7              Calculate $h_{i+M}$ by $R_i' \geq R_{th}$;
8              $E_i^{comp} = \frac{W_i}{f_j} P_j + h_{i+M} \frac{W_i}{f_{j'}} P_{j'}$;
9              if $E_i^{comp} < E_{i,\min}^{comp}$ then
10                 $E_{i,\min}^{comp} = E_i^{comp}$;
11                 $B[i] = j$;
```

Since the total number of tasks is $2M$ and each core of a node has L V/F levels, the complexity is $O(ML)$.

2. To deal with problem (21), we propose a task layer classification method to determine a sequence Q to perform task allocation and scheduling. Note that the number of original and duplicated tasks is M^h; in the worst case, we have $2M$. In addition, a recursive method is used to determine the task layer according to the task dependency. For each task τ_i, the height of the recursion tree is $2M$. Since the recursive method has a logarithmic growth with problem size [25], and this method is applied for $2M$ tasks, the complexity of task layer classification is $O(Mlog(M))$. Then, based on the task sequence Q, we determine the allocation, scheduling, and communication of tasks, i.e., the values of variables $\boldsymbol{q} = [q_{id}]_{2M\times NR}$, $\boldsymbol{x} = [x_{ik}]_{2M\times N}$, and $\boldsymbol{c} = [c_{\beta\gamma\rho}]_{N\times N\times C}$ in problem (21). Note that each task τ_i has at most $2M-1$ predecessors, and the task number is $2M$. Moreover, we have N nodes, each node has R cores, and the number of communication paths is C. The complexity is $O(M^2NRC)$. Based on the above analysis, the complexity of Algorithm 2 is $O(Mlog(M) + M^2NRC)$.
3. Algorithm 3 is used to adjust the V/F levels $\boldsymbol{y} = [y_{il}]_{M\times L}$ during the task allocation and scheduling of Algorithm 2. In the worst case, the number of tasks that violate the time constraint (11) in Algorithm 2 is M^h, where $M^h \leq 2M$. For each task τ_i with $t_i^e > H$, we calculate its minimum V/F level (v_{l+a}, f_{l+a}) and find the available V/F levels $\{(v_{l+a}, f_{l+a}), \dots, (v_L, f_L)\}$, where the maximum set size is $L-1$. For each V/F level, we calculate the task reliability R_i' and computation energy E_i^{comp}. Since the complexity FATD is $O(ML)$, i.e., similar to Algorithm 1, Algorithm 3 has complexity $O(ML)$.

Note that the number of tasks is much larger than that of V/F levels, cores, and data paths, i.e., $M \gg L$, $M \gg R$, and $M \gg C$. Without Algorithm 3, the total time complexity of our method is $O(ML + Mlog(M) + M^2NRC)$, and can be simplified as $O(Mlog(M) + M^2N)$. With Algorithm 3, the total time complexity is $O(ML + Mlog(M) + M^3LNRC)$, and can be simplified as $O(Mlog(M) + M^3N)$.

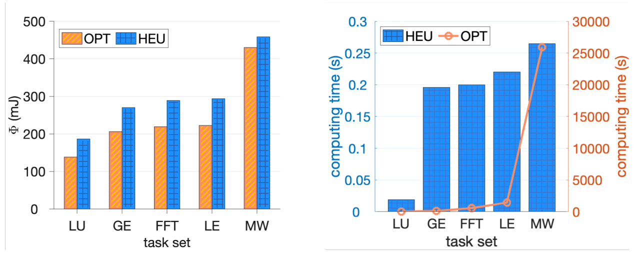
Table 2
Experimental set-up

Processor parameters					
v_l (V)	0.65	0.7	0.75	0.8	0.85
f_l (GHz)	1.01	1.26	1.53	1.81	2.10
P_l (mW)	184.9	266.7	370.4	498.9	655.5
P_l^s (mW)	246	290.1	340.3	397.6	462.7
P_0^s (μW)	80				
Task parameters					
$W_i \in [4\times10^7, 6\times10^8]$			$s_{ij} = 1$		
Reliability parameters					
$\lambda = 10^{-6}$		$d = 5$		$R_{th} = 0.998$	
Energy and Time Parameters					
$E_k^{ini} = E_{k,temp}^{comm} + E_{k,temp}^{comp}$					
$E_{k,temp}^{comm} = 2M \max_{\forall\beta,\gamma,\rho}\{e_{\beta\gamma k\rho}\}$			$E_{k,temp}^{comp} = 2M \max_{\forall i,l}\left\{\frac{W_i}{f_l}P_l\right\}$		
$D_i = \alpha \times t_{i,temp}^{comp}$					
$H = \sum_{i\in CPT}(t_{i,temp}^{comm} + t_{i,temp}^{comp})$					
$t_{i,temp}^{comm} = (t_{i,\min}^{comm} + t_{i,\max}^{comm})/2$			$t_{i,temp}^{comp} = (t_{i,\min}^{comp} + t_{i,\max}^{comp})/2$		
$t_{i,\min}^{comm} = \min_{\forall\beta,\gamma,\rho}\{t_{\beta\gamma\rho}\}$			$t_{i,\max}^{comm} = \max_{\forall\beta,\gamma,\rho}\{t_{\beta\gamma\rho}\}$		
$t_{i,\min}^{comp} = \min_{\forall l}\left\{\frac{W_i}{f_l}\right\}$			$t_{i,\max}^{comp} = \max_{\forall l}\left\{\frac{W_i}{f_l}\right\}$		

5. Experimental Evaluation

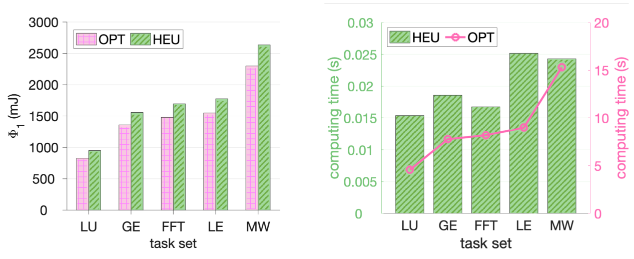
5.1. Simulation Setup

We perform extensive experiments to evaluate the performance and effectiveness of the proposed task mapping method. For the experimental setup, the processor power parameters (e.g., v_l, f_l, and P_l) are adopted from [26, 23]. The DAG task parameters (e.g., W_i and s_{ij}) and the reliability parameters (e.g., λ, d, and R_{th}) are adopted from [35, 33]. Table 2 summarizes the time and energy constraints settings. For node energy supply, we introduce a temporary communication energy $E_{k,temp}^{comm}$ and a temporary computation energy $E_{k,temp}^{comp}$, and set $E_k^{ini} = E_{k,temp}^{comm} + E_{k,temp}^{comp}$, where $E_{k,temp}^{comm}$ and $E_{k,temp}^{comp}$ are the maximum energy required to transmit the data for all the tasks $\{\tau_1, \dots, \tau_{2M}\}$ and execute these tasks, respectively. Similarly, we introduce a temporary computation time $t_{i,temp}^{comp}$ for the relative deadline $D_i = \alpha \times t_{i,temp}^{comp}$, where $t_{i,\min}^{comp}$ and $t_{i,\max}^{comp}$ in $t_{i,temp}^{comp}$ are the minimum and maximum time required to execute τ_i, respectively, and α is a turned parameters. Since the tasks are dependent, to have a schedulable task set, the scheduling horizon is set to $H = \sum_{i\in CPT}\left(t_{i,temp}^{comm} + t_{i,temp}^{comp}\right)$, where CPT is the set of the tasks on the critical path, $t_{i,temp}^{comm}$ is the temporary communication time for task τ_i, $t_{i,\min}^{comm}$ and $t_{i,\max}^{comm}$ are the minimum and maximum data communication time for τ_i, respectively. Note that different platforms and applications



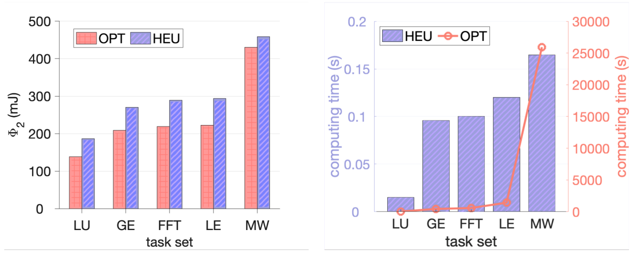
(a) Energy consumption. (b) Computation time.

Figure 4: The solution and the time of PP with OPT and HEU methods.



(a) Energy consumption. (b) Computation time.

Figure 5: The solution and the time of FATD with OPT and HEU methods.



(a) Energy consumption. (b) Computation time.

Figure 6: The solution and the time of TAPS with OPT and HEU methods.

lead to different parameter values for the task mapping problem (12). However, the problem structure under different parameter values is the same; thus, the proposed methods are still applicable.

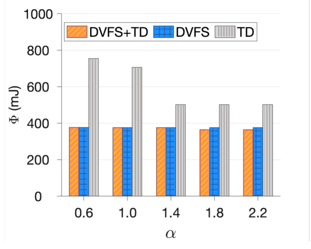
For the evaluation metrics, we compare the solution quality and computation time of PP (12) achieved by the optimal method (OPT) and the heuristic method (HEU) under different realistic applications, i.e., LU decomposition (LU), Gaussian elimination (GE), fast Fourier transform (FFT), Laplace equation (LE), and Montage workflows (MW) [30, 2, 9]. Furthermore, we compare the results with different task reliability schemes: 1) the combination of DVFS and task duplication (DVFS+TD), 2) only using DVFS (DVFS) [1], and 3) only using task duplication (TD) [5]. On this basis, we compare the results with single-path (SP) [20] and multi-path (MP) data routing. Finally, we evaluate the behavior of the methods under different node parameters, e.g., communication and computing energy ratios and processor V/F level intervals. The simulations are carried out on a 32-core CPU and 64 GB RAM server, and the algorithms are implemented through MATLAB 2021a and Gurobi 9.5.1.

5.2. Performance Evaluation

Fig. 4 compares the solution quality and the computation time achieved by the OPT and HEU methods, where the task numbers of LU, GE, FFT, LE, and MW are M = 9, 14, 15, 16, and 24, respectively. The results show that as the task number M increases, the computation time of OPT and HEU increases since more variables and constraints are involved in the task mapping problem. However, HEU can reduce 99.9% computation time compared with OPT; the computation time of HEU is negligible, as it is usually within 0.5 s. The results also show that HEU has a 24.3% performance loss compared with OPT.

Note that FATD is an INLP problem, and TAPS is a MINLP problem. These nonlinear subproblems can be linearized through *Lemma* 3.2 and then optimally solved by the Gurobi solver. Fig. 5 and Fig. 6 compare the solution quality and the computation time achieved by the optimal methods and our design methods summarized in Algorithm 1 and Algorithm 2, when linearization takes place for FATD and TAPS. The results show that our methods can find feasible solutions to FATD and TAPS quickly with 12.7% and 24.1% higher energy consumption than the optimal solutions. The complexity of an optimization problem is highly related to the problem size. The proposed heuristic decomposes the joint optimization problem into two subproblems, FATD and TAPS, with fewer variables, constraints, and coupled nonlinear items, and then solves them in sequence with low computational complexity algorithms. Since our heuristics can largely reduce computation time with an acceptable performance loss, it is suitable for large-scale networked systems.

Fig. 7 shows the influence of different task reliability schemes (i.e., DVFS, TD, and DVFS+TD) on energy consumption (objective function of PP), where the task number $M \in [5, 200]$ and the time parameter $\alpha \in [0.6, 2.2]$. Note that the task relative deadline D_i is controlled by α; the smaller α, the shorter D_i. The results show that with the time parameter α increasing, the energy consumptions of DVFS, TD, and DVFS+TD decrease. This is because the task mapping aims to reduce the energy consumption of the nodes. With the time constraint relaxed, the lower V/F level and fewer replicas can be used to satisfy the reliability constraint. However, compared with DVFS and TD, DVFS+TD has a lower energy consumption. To meet the task reliability constraint, DVFS will use a higher V/F level to execute the tasks, while TD will generate more replicas, thus leading to more energy consumption during task computation and data transmission processes. However, with the introduction of constraint (6), DVFS+TD combines the benefits of DVFS and task duplication, i.e., it can optimize the variables y_{il} and h_i at the same time, to reduce energy consumption by


(a) Change of Φ with parameter α.

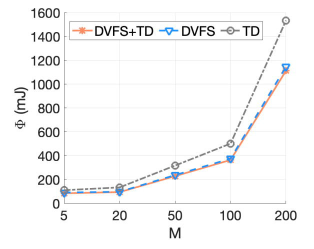
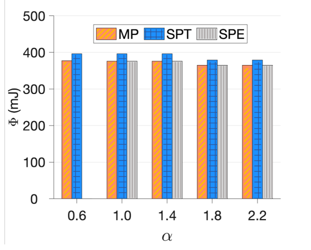

(b) Change of Φ with parameter M.

Figure 7: Energy consumption comparison with different task reliability schemes and time parameters.


(a) Change of Φ with parameter α.

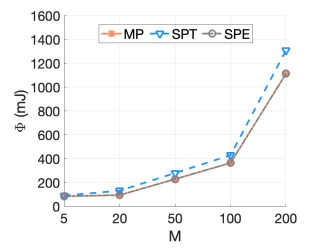
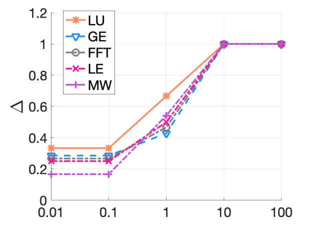

(b) Change of Φ with parameter M.

Figure 8: Energy consumption comparison with different time parameters and data routing schemes.

using lower V/F levels and fewer replicas. Therefore, combining DVFS and task duplication enhances task reliability in network environments.

Fig. 8 compares the energy consumption with SP and MP data routing schemes, where SPE and SPT represent the single-path data routing with the energy-oriented and time-oriented data paths, respectively. The results show that compared with SP, MP has a lower energy consumption. This is because SP is a particular case of MP, i.e., the variable $c_{\beta\gamma\rho}$ is fixed in MP. As another dimension of variables is introduced by MP, it has a larger feasible region, and thus, it has a lower energy consumption for a minimization problem. The result also shows that the task mapping problem with MP is more feasible. When the time parameter α is small, the time constraint (10) becomes stricter. The task mapping problem may be infeasible for SP when $\alpha = 0.6$. However, the problem is feasible with MP. Therefore, MP is suitable for the task mapping problem with multiple constraints. In fact, DVFS+TD also plays a similar role as MP since DVFS and TD can be treated as the special cases of DVFS+TD with $h_i = 0$ and $h_i = 1$ $(M \le i \le 2M)$, respectively. By jointly considering DVFS, task duplication, and multipath data routing during task mapping, we can better use system resources to improve energy efficiency.

Fig. 9(a) evaluates the influence of communication and computation energy on task-to-node allocation. To describe the relationship between the communication and computation energy consumption, we introduce two parameters $\mu = E_k^{comm}/E_k^{comp}$ and $\Delta = M_{\max}/M_{total}$. In μ, we set $E_k^{comm} = \max_{\forall\beta,\gamma,\rho}\left\{E_{\beta\gamma k\rho}\right\}$ and $E_k^{comp} = \max_{\forall i,l}\left\{\frac{W_i}{f_l}P_l\right\}$. Hence,


(a) Influence of energy ratio μ on task allocation.

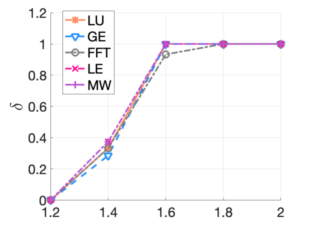

(b) Influence of V/F level interval ϵ on task duplication.

Figure 9: The influence of system communication and computation parameters on task allocation and duplication.

for a node θ_k, the larger the value of μ is, the more energy is consumed to transmit data, compared with task execution. Let M_k denote the total number of tasks allocated to θ_k. In Δ, $M_{\max} = \max_{\forall k}\{M_k\}$ is the maximum number of tasks that are allocated to a node, and $M_{total} = \sum_{\forall k} M_k$ is the total number of allocated tasks. Thus, Δ represents the rate of task distribution among the nodes. The higher the value of Δ, the more concentrated of tasks allocated to the nodes. Fig. 9(a) shows that Δ increases with μ. This is because when the value of μ is small, the communication energy is smaller than the computation energy. To balance the energy consumption among the nodes, the tasks are prone to be distributed to different nodes and thus lead to a small Δ. With μ increasing, the dependent tasks are prone to be allocated to the same nodes to reduce the communication energy, leading to the increase of Δ. Therefore, the communication and computation characteristics of the nodes will affect the task allocation results.

Fig. 9(b) evaluates the influence of the voltage/frequency interval on the task duplication decision. Since $\sum_{l\in\mathcal{L}} y_{il}\frac{W_i}{f_l}P_l$ is the energy required to execute task τ_i, the parameter $\epsilon = \max_{\forall l}\left\{P_l/f_l\right\}/\min_{\forall l}\left\{P_l/f_l\right\}$ represents the range of V/F level gap. The larger ϵ, the difference between the V/F levels is more apparent. On the other hand, we introduce a parameter $\delta = M_d/M$ to represent the rate of task duplication, as M_d is the number of duplicated tasks, and M is the number of original tasks. Fig. 9(b) shows that δ increases with ϵ. This is because when ϵ is small, the gap between different V/F levels is small. To satisfy the reliability constraint (7) and the time constraints (10) and (11), executing two tasks (the original and duplicated tasks) with low frequency will consume more energy than executing one task (the original task) with high frequency. Tasks are prone to being executed with a high V/F level without task duplication and thus lead to a small δ. However, with ϵ increasing, task duplication is more energy efficient to satisfy multiple constraints, increasing the duplicated task number M_d. Therefore, the processor parameters will influence the V/F selection and replica number of the DVFS+TD scheme.

6. Related Work

This section discusses the related work regarding task mapping inside a platform with multiple cores and among multiple nodes in networked systems, targeting energy under multiple constraints.

Regarding the approaches focusing on multicore mapping inside a single platform, approaches exist that apply DVFS to reduce energy consumption. For instance, frame-based independent tasks are considered in [14], and dependent tasks with DVFS and Dynamic Power Management (DPM) are studied in [6]. The task allocation problem is formulated as an Integer Linear Programming (ILP) and solved by a decomposition-based heuristic method [14], while the task mapping problem is formulated as a MILP and solved by the CPLEX solver [6]. Task migration mechanism is employed in [28] to improve task execution efficiency of ARM big.LITTLE platforms, where one task is divided into several subtasks executed on the cores with heterogeneous characteristics to balance task execution time and energy. These works consider the influence of DVFS on energy efficiency, neglecting the impact on task reliability due to low voltage/frequency levels.

Other works focus on fault-tolerant techniques, which incur significant time and energy overheads to the multicore platforms. For instance, task reliability is optimized through DVFS during task mapping [22] because task reliability follows a Poisson distribution regarding working frequency and supply voltage. Usually, higher task reliability requires a higher voltage/frequency level. Task replication is used to enhance task reliability, such as task duplication [36]. Since it is unlikely that the execution of task replicas on different cores fails, task reliability increases with the number of replicas. For instance, many real applications, e.g., AGV and UAG [24], use more than two replicas to satisfy the given reliability threshold based on safety standards. However, it introduces more tasks for execution. DVFS and task duplication are combined to reduce computation energy under the reliability constraints for the data bus-based [7, 11] and NoC-based [19] multicore platforms, as it optimizes the number of tasks that require duplication, i.e., from full duplication to partial duplication. The above studies mainly focus on a single multicore platform without considering the communication between different multicore platforms; thus, task mapping will not influence the communication costs between multicore platforms.

Regarding the mapping approaches focusing on multiple nodes of the networked systems, they consider computation and communication costs [8, 32], where the nodes with embedded systems as process units, and the nodes are connected to form networked systems. For instance, task allocation mechanisms with and without DVFS are considered and compared to reduce computation and communication energy under timing constraints [21]. DVFS and multipath data routing are used to optimize the mapping process of dependent tasks on the nodes to balance the energy consumption of the nodes under time and energy constraints [17]. However, task reliability issues are not considered in [21, 17]. The problems of allocating and scheduling dependent tasks on the nodes of the networked system are studied in [18, 37], where task allocation and duplication [18] and task scheduling and routing [37] are jointly optimized to enhance task reliability. The target platforms of the above studies are single-core. The nodes with multicore platforms collaborate with the cloud to execute IoT applications [16], e.g., federated learning [34], where task mapping and offloading are performed on the nodes and the cloud, respectively, so as to reduce the computation and communication energy of nodes under timing constraints. Task reliability issues are not considered in [34, 16] since the cloud is assumed to have rich computation resources. Compared with the above works, we consider mapping dependent and real-time tasks on the networked system nodes with a multicore architecture. During the task mapping process, DVFS, task duplication, and multipath data routing are jointly optimized to improve energy efficiency under time, reliability, and dependency constraints.

7. Conclusion

This work studies the MINLP-based mapping problem of dependent tasks over a networked system, whose nodes are realized with multicore embedded platforms enhanced with DVFS. To balance the system's energy consumption under real-time response, task dependency and reliability constraints, task allocation, frequency assignment, task duplication, and path selection decisions are jointly optimized. We find the optimal solution through a linearization method and design a decomposition-based heuristic method with reduced computation time. The results show that our task mapping method can optimize the system's computation and communication energy consumption under multiple constraints and outperform other methods regarding energy efficiency, task reliability, and schedulability.

References

[1] Ali, H., Tariq, U.U., Liu, L., Panneerselvam, J., Zhai, X., 2019. Energy optimization of streaming applications in IoT on NoC-based heterogeneous MPSoCs using retiming and DVFS, in: IEEE SmartWorld, pp. 1297–1304.

[2] Arabnejad, H., Barbosa, J.G., 2013. List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans. Parallel Distrib. Syst. 25, 682–694.

[3] Billet, B., Issarny, V., 2014. From task graphs to concrete actions: a new task mapping algorithm for the future Internet of things, in: IEEE International Conference on Mobile Ad Hoc and Sensor Systems, pp. 470–478.

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

[5] Chatterjee, N., Paul, S., Chattopadhyay, S., 2017. Fault-tolerant dynamic task mapping and scheduling for network-on-chip-based multicore platform. ACM Trans. Embed. Comput. Syst. 16, 1–24.

[6] Chen, G., Huang, K., Knoll, A., 2014. Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination. ACM Trans. Embed. Comput. Syst. 13, 1–21.

[7] Cui, M., Kritikakou, A., Mo, L., Casseau, E., 2021. Fault-tolerant mapping of real-time parallel applications under multiple DVFS

schemes, in: Proc. IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 387–399.

[8] Dai, X., Burns, A., 2020. Period adaptation of real-time control tasks with fixed-priority scheduling in cyber-physical systems. Journal of Syst. Architect. 103, 101691.

[9] Deelman, E., Singh, G., 2005. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Sci. Prog. 13, 219–237.

[10] Dey, S., Isuwa, S., Saha, S., Singh, A.K., McDonald-Maier, K., 2022. CPU-GPU-memory DVFS for power-efficient MPSoC in mobile cyber physical systems. Future Internet 14.

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

[12] He, J., Xiao, Y., Bogdan, C., Nazarian, S., Bogdan, P., 2021. A design methodology for energy-aware processing in unmanned aerial vehicles. ACM Trans. Des. Autom. Electron. Syst. 27, 1–20.

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

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

[15] Lin, X., Wang, Y., Xie, Q., Pedram, M., 2015. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. IEEE Trans. Serv. Comput. 8, 175–186.

[16] Liu, J., Liu, C., Wang, B., Gao, G., Wang, S., 2022. Optimized task allocation for IoT application in mobile-edge computing. IEEE Internet Things J. 9, 10370–10381.

[17] Mo, L., Kritikakou, A., Sentieys, O., Cao, X., 2021. Real-time imprecise computation tasks mapping for DVFS-enabled networked systems. IEEE Internet Things J. 8, 8246–8258.

[18] Mo, L., Zhou, Q., Kritikakou, A., Cao, X., 2023. Energy optimized task mapping for reliable and real-time networked systems. ACM Trans. Sen. Netw. 19.

[19] Mo, L., Zhou, Q., Kritikakou, A., Liu, J., 2022. Energy efficient, real-time and reliable task deployment on NoC-based multicores with DVFS, in: Proc. Design, Automation & Test in Europe Conference & Exhibition, pp. 1347–1352.

[20] Namazi, A., Abdollahi, M., Safari, S., Mohammadi, S., 2017. A majority-based reliability-aware task mapping in high-performance homogenous NoC architectures. ACM Trans. Embed. Comput. Syst. 17, 1–31.

[21] Pathak, A., Prasanna, V., 2010. Energy-efficient task mapping for data-driven sensor network macroprogramming. IEEE Trans. Comput. 59, 955–968.

[22] Peng, J., Li, K., Chen, J., Li, K., 2022. Reliability/performance-aware scheduling for parallel applications with energy constraints on heterogeneous computing systems. IEEE Trans. Sustain. Comput. 7, 681–695.

[23] Qin, Y., Zeng, G., Kurachi, R., Matsubara, Y., Takada, H., 2019. Execution-variance-aware task allocation for energy minimization on the big.LITTLE architecture. Sustainable Computing: Informatics and Systems 22, 155–166.

[24] Rice, L., Cheng, A., 1999. Timing analysis of the X-38 space station crew return vehicle avionics, in: Proc. IEEE Real-Time Technology and Applications Symposium, pp. 255–264.

[25] Schönhage, A., 1982. The fundamental theorem of algebra in terms of computational complexity. Manuscript. Univ. of Tübingen, Germany .

[26] Singh, J., Mangipudi, B., 2012. Restricted duplication based MILP formulation for scheduling task graphs on unrelated parallel machines, in: IEEE International Symposium on Parallel Architectures, Algorithms and Programming, pp. 202–209.

[27] Tariq, U.U., Ali, H., Liu, L., Hardy, J., Kazim, M., Ahmed, W., 2021. Energy-aware scheduling of streaming applications on edge-devices in IoT-based healthcare. IEEE Trans. Green Commun. Networking 5, 803–815.

[28] Thammawichai, M., Kerrigan, E., 2018. Energy-efficient real-time scheduling for two-type heterogeneous multiprocessors. Real-Time Syst. 54, 132–165.

[29] Tong, E., Niu, W., Tian, Y., Liu, J., Baker, T., Verma, S., Liu, Z., 2021. A hierarchical energy-efficient service selection approach with QoS constraints for Internet of things. IEEE Trans. Green Commun. Netw. 5, 645–657.

[30] Topcuoglu, H., Hariri, S., Wu, M., 2002. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. 13, 260–274.

[31] Wei, T., Zhou, J., Cao, K., Cong, P., Chen, M., Zhang, G., Hu, X., Yan, J., 2018. Cost-constrained QoS optimization for approximate computation real-time tasks in heterogeneous MPSoCs. IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 37, 1733–1746.

[32] Wu, P., Fu, C., Wang, T., Li, M., Zhao, Y., Xue, C., Han, S., 2021. Composite resource scheduling for networked control systems, in: IEEE Real-Time Systems Symposium, pp. 162–175.

[33] Xie, G., Chen, Y., Xiao, X., Xu, C., Li, R., Li, K., 2017. Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems. IEEE Trans. Sustain Comput. 3, 167–181.

[34] Yao, J., Ansari, N., 2021. Enhancing federated learning in fog-aided IoT by CPU frequency and wireless power control. IEEE Internet Things J. 8, 3438–3445.

[35] Yu, H., Ha, Y., Veeravalli, B., 2013. Quality-driven dynamic scheduling for real-time adaptive applications on multiprocessor systems. IEEE Trans. Computers 62, 2026–2040.

[36] Zhou, J., Sun, J., Zhou, X., Wei, T., Chen, M., Hu, S., Hu, X., 2019. Resource management for improving soft-error and lifetime reliability of real-time MPSoCs. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 38, 2215–2228.

[37] Zhou, Y., Samii, S., Eles, P., Peng, Z., 2021. Reliability-aware scheduling and routing for messages in time-sensitive networking. ACM Trans. Embed. Comput. Syst. 20, 1–24.

Lei Mo is an associate professor with the School of Automation, Southeast University, Nanjing, China. He received the B.S. degree from the College of Telecom Engineering and Information Engineering, Lanzhou University of Technology, Lanzhou, China, in 2007, and the Ph.D. degree from the College of Automation Science and Engineering, South China University of Technology, Guangzhou, China, in 2013. From 2013 to 2015, he was a research fellow with the Department of Control Science and Engineering, Zhejiang University, China. From 2015 to 2017, he was a research fellow with INRIA Nancy–Grand Est, France. From 2017 to 2019, he was a research fellow with INRIA Rennes–Bretagne Atlantique, France. His research interests include networked estimation and control in wireless sensor and actuator networks, cyber-physical systems, task mapping and resource allocation in embedded systems.

Jingyi Zhang received her B.S. degree in automation engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2022. She is currently working towards the M.S. degree in Control Science and Engineering at Southeast University, Nanjing, China. Her current research interests include embedded and real-time systems, mixed-criticality systems, and networked systems.

Minyu Cui is a research fellow with the Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden. She received the B.E. degree in optical information science and technology and the M.S. degree in electronics and communication engineering from Shandong University in 2013 and 2018, respectively, and the Ph.D. degree from École Normale Supérieure de Rennes at IRISA/Inria Rennes in 2022. Her research interests include algorithm design, resource optimization, task mapping, multicore architecture, fault-tolerance, computing systems, and real-time systems.

Xiaoyong Yan received the PhD degree in computer science and technology from the Nanjing University of Science and Technology, Nanjing, China, in 2013. He is currently an associate professor with the Nanjing University of Posts and Telecommunications. His research interests include wireless networks and Internet of Things.

Shuang Wang received the BSc from the College of Sciences from Nanjing Agricultural University in 2015 and the PhD degree from the School of Computer Science and Engineering, Southeast University, Nanjing, China in 2020. She was a visiting PhD student with the School of Computing, Macquarie University, Sydney, Australia, from 2019 to 2020, and a post-doctoral research fellow from 2020 to 2021. She is currently a lecturer with Southeast University. Her main research interests focus on service computing, big data analytics, and truth discovery.

Xiaojun Zhai is a Reader in the Embedded Intelligent Systems Laboratory at the University of Essex. His research interests include the design and implementation of digital image and signal processing algorithms, custom computing using FPGAs, embedded systems, and hardware/software co-design.