HAL
open science

# Energy-Quality-Time Optimized Task Mapping on DVFS-enabled Multicores

Lei Mo, Angeliki Kritikakou, Olivier Sentieys

## To cite this version:

Lei Mo, Angeliki Kritikakou, Olivier Sentieys. Energy-Quality-Time Optimized Task Mapping on DVFS-enabled Multicores. ESWEEK 2018 - Embedded Systems Week, Sep 2018, Torino, Italy. pp.1-11. hal-01941764

**HAL Id: hal-01941764**

**https://inria.hal.science/hal-01941764**

Submitted on 2 Dec 2018

**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-Quality-Time Optimized Task Mapping on DVFS-enabled Multicores

Lei Mo, *Member, IEEE,* Angeliki Kritikakou, and Olivier Sentieys, *Member, IEEE*

***Abstract*—Multicore architectures have great potential for energy-constrained embedded systems, such as energy-harvesting wireless sensor networks. Some embedded applications, especially the real-time ones, can be modeled as imprecise computation tasks. A task is divided into a mandatory subtask that provides a baseline Quality-of-Service (QoS) and an optional subtask that refines the result to increase the QoS. Combining dynamic voltage and frequency scaling, task allocation and task adjustment, we can maximize the system QoS under real-time and energy supply constraints. However, the nonlinear and combinatorial nature of this problem makes it difficult to solve. This work first formulates a mixed-integer non-linear programming problem to concurrently carry out task-to-processor allocation, frequency-to-task assignment and optional task adjustment. We provide a mixed-integer linear programming form of this formulation without performance degradation and we propose a novel decomposition algorithm to provide an optimal solution with reduced computation time compared to state-of-the-art optimal approaches (22.6% in average). We also propose a heuristic version that has negligible computation time.**

***Keywords*—Multicore architectures, task mapping, real-time and energy constraints, QoS, MILP, problem decomposition**

## I. Introduction

The increasing performance requirements of applications have pushed the embedded systems towards multicore architectures. Within thirty years, the code size of automotive, space and avionics applications has significantly increased [1]. Multicore architectures provide significant Space, Weight and Power savings (SWaP) while offering massive computing capabilities compared with single core processors. They are also capable of integrating diverse applications on the same platform [2], [3].

In some application domains, e.g., audio/video streaming, radar tracking, and control system design, less accurate results computed before the deadline are preferable than accurate, but too late, results [4]. This statement holds because a real-time application has to provide a result before a given deadline. When not enough time is available, approximate results are acceptable as long as the baseline Quality-of-Service (QoS) is satisfied and the results are provided in time [5]. For instance, in audio/video streaming, frames with a lower quality are better than missing frames. In radar tracking, an estimation of target's location in time is better than an accurate location arriving too late. In control loops, an approximate result produced by a control law is more preferable as long as the controlled system, e.g., cruise control system, remains stable. In these domains, the applications can be modeled as *Imprecise Computation* (IC) tasks [6], where a task is logically decomposed into a mandatory subtask and an optional subtask. All the mandatory subtasks must be completed before the deadline to have an acceptable result, while the optional subtasks can be left incomplete at the cost of reduced quality. The QoS of such systems increases with the longer execution of the optional subtasks.

At the same time, the energy consumption has become an important concern, especially for the systems with limited energy budget, such as battery powered devices and energy-harvesting systems. Dynamic Voltage and Frequency Scaling (DVFS) is a technique that can manage both voltage and frequency, and, thus, control the energy and the time required to execute the tasks. By adequately allocating the tasks onto the processors and deciding the voltage/frequency of the processor for each task execution, the QoS can be further improved under the same energy supply and real-time constraints.

The majority of the task mapping approaches on multicores, e.g., [7]–[12], has been conducted on how to allocate the tasks and/or decide the voltage/frequency in order to reduce the energy consumption of the system and guarantee the real-time constraints, i.e., energy-aware task mapping. However, most of these works focus only on the execution of the mandatory tasks. Some approaches, e.g., [4], [13]–[18], aim to maximize QoS by exploring the execution of the optional subtasks, with energy supply and real-time constraints, i.e., QoS-aware task mapping. However, the aforementioned approaches focus on different contexts compared to the problem considered in this paper, as summarized in Table I. Besides single-objective optimization, other existing approaches focus on bi-objective optimization, e.g., minimizing the energy consumption and maximizing the QoS, but without restrictions on the energy supply [19], [20]. In addition, the task mapping problems on multicore platforms usually have an $\mathcal{NP}$–hard complexity [21]. Therefore, most of these QoS-aware task mapping approaches focus on heuristics to find feasible solutions [14]–[17], [19], [20].

Complementary to the state-of-the-art, our work jointly optimizes task-to-processor allocation, frequency-to-task assignment and optional subtask adjustment, under energy supply and real-time constraints, with an objective of maximizing the QoS. Two algorithmic solutions are proposed: 1) an optimal approach, named *Optimal Joint DVFS Task Mapping* (OJTM), with lower computational complexity than the state-of-the-art optimal approaches, and 2) a polynomial-time complexity algorithm, named *Heuristic Joint DVFS Task Mapping* (HJTM), with better quality results compared with the state-of-the-art heuristics. Heuristics are of great practical significance, since they are able to find feasible solutions in a short time. However, it is also important to find the optimal solution. Only by doing so, we can find out how far the obtained feasible solution is from the optimal one, and how to improve the heuristics based on the optimal solution.

L. Mo. A. Kritikakou, and O. Sentieys are with Univ Rennes, INRIA, IRISA, CNRS, 35042 Rennes Cedex, France. E-mail: lei.mo@inria.fr, angeliki.kritikakou@irisa.fr, olivier.sentieys@irisa.fr.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis 2018 and appears as part of the ESWEEK-TCAD special issue.

TABLE I. TASK MAPPING METHOD

| | | *Energy-aware* | | | | | | *QoS-aware* | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | [10] | [11] | [12] | [9] | [7] | [8] | [4] | [13] | [20] | [19] | [18] | [17] | [14] | [15] | [16] | OJTM | HJTM |
| *Variables* | Frequency-to-task | √ | √ | √ | | √ | √ | | √ | √ | √ | | | √ | √ | | √ | √ |
| | Task-to-processor | √ | | √ | √ | √ | | √ | | √ | √ | √ | √ | | | √ | √ | √ |
| | Task adjustment | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *Objective* | Max. QoS | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| | Min. Energy | √ | √ | √ | √ | √ | √ | | | √ | √ | | | | | | | |
| *Constraints* | Real-time | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| | Energy | | | | | | | | √ | | | √ | √ | √ | √ | √ | √ | √ |
| *Solution* | Optimal | √ | √ | | | | | √ | √ | | | √ | | | | | √ | |
| | Non-optimal | | | √ | √ | √ | √ | | | √ | √ | | √ | √ | √ | √ | | √ |

### A. Related Work

*1) Energy-aware Task Mapping:* Existing works that focus on the pure energy-aware task mapping problem aim at minimizing the energy consumption under resource and application constraints, e.g., [7]–[12]. DVFS is widely used in energy-aware task mapping. When voltage level is discrete, the problems are usually formulated as Integer Programming (IP), e.g., [7]–[9]. To efficiently solve the IP problems, a hybrid Genetic Algorithm (GA) is proposed in [7], a polynomial-time complexity two-step heuristic is designed in [8], and the IP problem is relaxed to a Linear Programming (LP) in [9]. Combining DVFS and Dynamic Power Management (DPM), a Mixed-Integer Linear Programming (MILP)-based task mapping problem is considered in [10] and the problem is solved using the CPLEX optimization solver. When the voltage level is continuous, a convex problem is formulated in [11] and the problem can be solved by using polynomial-time methods. In [12], Mixed-Integer Non-Linear Programming (MINLP) is used to formulate the task mapping problem. The problem is relaxed to an MILP by linear approximation and solved by Branch and Bound (B&B) method [22]. However, the tasks in aforementioned approaches are not modifiable, and, thus, no exploration of the QoS improvement through the optional subtasks adjustment is considered.

*2) QoS-aware Task Mapping:* Other works consider the QoS-aware task mapping problem using the IC task model and having a goal to maximize the QoS under a set of real-time and/or energy supply constraints, e.g., [4], [13]–[20]. The target platforms considered in [13], [14] are single core platforms. Therefore, the task allocation problem is not taken into account. Although the works in [4], [15]–[20] target at multicore platform, some assumptions are made during the problem formulation. More precisely, the task-to-processor allocation is fixed and given in advance for all the tasks in [15], while in [4], [16]–[18] each processor has a predefined frequency, and in [19], [20] the energy supply constraints are not taken into account since the aim is to maximize the QoS as well as to minimize the energy consumption. When taking multiple system requirements into account, the complex coupling between the optimization variables makes the problem difficult to solve, especially when the coupling is non-linear and non-convex. The methods that used to solve the aforementioned problems can be classified into two main classes. The first class includes the methods based on heuristics, e.g., [14]–[17], [19], [20]. The second class includes the methods that always produce an optimal solution, e.g., [4], [13], [18].

### B. Contributions

This paper solves the following problem: *Given a DVFS-enabled multicore platform and a set of IC tasks, we determine which processor should the task be executed on as well as the frequency of each task, such that the system QoS is maximized while guaranteeing the system constraints, i.e., energy supply and task deadline*. Hence, we need to decide task-to-processor allocation, frequency-to-task assignment and task adjustment. Our main contributions are summarized as follows:

1) We propose an MINLP formulation of the QoS-aware mapping problem of IC tasks on DFVS multicore platforms under a set of energy supply and real-time constraints. The task-to-processor allocation, frequency-to-task assignment, and task adjustment are optimized simultaneously. Solving these correlated subproblems separately may lead to non-optimal solutions. Then, applying the variable replacement method, the $\mathcal{NP}$–hard joint-design problem is safely transformed to an MILP. We prove that the proposed linearization does not cause performance degradation, compared with the existing linear approximation methods [12].
2) We design an OJTM solving approach based on Benders decomposition [23]. The MILP-based task mapping problem is divided into two subproblems with simpler structures, i.e., with less constraints and variables. The first subproblem is an Integer Linear Programming (ILP) and it is responsible for the task-to-processor allocation and the frequency-to-task assignment. The second subproblem is an LP and it is related to the task adjustment. The solution is found through iterations. During one iteration, the solution of the first subproblem is used as input to the second sub-problem. The solution of the second subproblem is incorporated to the first one by adding new constraints for the next iteration. We prove that by iterating these two subproblems, the OJTM is guaranteed to reach the global optimal solution.
3) We present a novel HJTM algorithm that reduces the complexity of OJTM by removing the iteration process. As the HJTM is a reduced version of OJTM, it is easy to be applied to similar MILP problems, compared with the existing heuristics. When the problem formulation is changed, existing heuristics usually have to be redeveloped, such as [16], [20].
4) We evaluate the performance of the proposed approaches by a set of extended experiments, where the proposed approaches are compared both in QoS and in computation time with the state-of-the-art optimal, stochastic and heuristic approaches. The obtained results show an average 22.6% reduction in the computation time for the OJTM compared with state-of-the-art optimal approaches. On the other hand, HJTM runs $\sim$100 times faster than OJTM with a cost of 26.3% QoS reduction. Moreover, compared with the state-of-

the-art heuristics, HJTM achieves an average 96.7% QoS improvement.

### C. Paper Organization

The remainder of this paper is organized as follows: Section II presents the system model and the problem formulation. Section III describes the OJTM algorithm to optimally map IC tasks to DVFS-enabled multicore platform and in Section IV we present the details of HJTM algorithm. Section V evaluates the performance of OJTM and HJTM algorithms through several simulations and Section VI concludes this study.

## II. MINLP Problem Formulation

### A. Task model

The applications of our target domain are modeled as IC tasks [6].

*Definition 2.1 (Imprecise computation task):* A task $\tau_i$ is defined as an imprecise computation task when it can be logically divided into a mandatory subtask and an optional subtask. The mandatory subtask executes $M_i$ cycles to generate the baseline QoS level. The optional subtask executes $o_i$ cycles to improve the QoS.

Consider a task set of $N$ independent IC tasks $\{\tau_1, \ldots, \tau_N\}$. Task $\tau_i$ is described by a tuple $\{o_i, M_i, O_i, D_i\}$, where $O_i$ is the upper bound on the number of possible optional cycles, i.e., $0 \leq o_i \leq O_i$. Therefore, the total length of task $\tau_i$, measured in execution cycles, is $M_i + o_i$. The exact number of mandatory cycles $M_i$ and maximum optional cycles $O_i$ of a task $\tau_i$ depends on the actual execution of the task on the multicore system [4], [5]. As we require to guarantee the deadline of the task, the $M_i$ and $O_i$ are measured in Worst Case Execution Cycles (WCECs) [14]. For each task $\tau_i$, there is a relative deadline $D_i$ before which both the mandatory and the optional subtasks of $\tau_i$ must be completed. The relative deadline is defined as the time interval between the start of a task and the time instance when the deadline occurs [24]. The scheduling horizon $H$ is given by the global period of the tasks [16]. Hence, the relative deadline $D_i$ cannot be greater that the scheduling horizon $H$, i.e., $D_i \leq H$.

### B. Energy model

We assume that the IC tasks run on a multicore platform that contains $M$ homogeneous processors $\{\theta_1, \ldots, \theta_M\}$. The processors can support DVFS and each processor $\theta_k$ has $L$ different discrete voltage/frequency levels $\{(v_1, f_1), \ldots, (v_L, f_L)\}$. We consider that the processor energy consumption is not only determined by the dynamic power, but also by the static power due to CMOS device feature scaling. The dynamic power by executing task $\tau_i$ is related to processor clock frequency $f_l$ and supply voltage $v_d$, expressed as $P_l^d = C_{eff} v_l^2 f_l$, where $C_{eff}$ is the average effective switching capacitance. For static power, we adopt $P_l^s = v_l K_1 e^{K_2 v_l} e^{K_3 v_{bs}} + |v_{bs}| I_j$ from [25]. The constants $K_1$, $K_2$ and $K_3$ depend on the processor technology. $I_j$ is the approximately constant junction leakage current. $v_{bs}$ is a reverse bias voltage used to reduce the leakage power and can be treated as constant [26]. Hence, the power of the processor under the voltage/frequency level $(v_l, f_l)$ is

$$P_l^c = P_l^s + P_l^d. \tag{1}$$

This model is widely adopted by the similarly published works [10], [15], [16].

For readability reasons in the rest of the manuscript, we use the term "frequency" instead of "voltage/frequency". When the frequency is determined, the corresponding voltage is also given. Examples of processor frequency-voltage characteristics are provided in Table II of the experimental section. In this paper, we consider inter-task DVFS [10], [14], i.e., the frequency of a processor stays constant for the entire execution of a task. Unlike the big.LITTLE platforms [27] that support task migration, we consider that a task starts execution on a processor and ends its execution on the same processor [17], [20]. We assume that the system is energy-constrained in the sense that it has a fixed energy budget $E_s$ that cannot be replenished during the scheduling horizon $H$ [16], [17]. Taking the available energy $E_s$ into account, the system operation can be divided into three states: 1) *Low*: the supplied energy $E_s$ is insufficient to execute all the mandatory cycles $\{M_1, \ldots, M_N\}$, 2) *High*: the supplied energy $E_s$ is sufficient to execute all the mandatory and optional cycles $\{M_1+O_1, \ldots, M_N+O_N\}$, and 3) *Medium*: all the mandatory cycles are ensured to finish, while not all the optional cycles can complete their executions. In this paper, we focus on medium state.

### C. Problem Formulation

The objective is to maximize the QoS subject to a set of real-time and energy constrains. In this context, we determine 1) on which processor should the task be executed (task-to-processor allocation), 2) what frequency should be used for the task (frequency-to-task assignment), and 3) how many optional cycles should be executed (task adjustment). Therefore, we define the following variables:

1) Binary variables

$$q_{ik} = \begin{cases} 1 & \text{if task } \tau_i \text{ runs on processor } \theta_k, \\ 0 & \text{otherwise.} \end{cases}$$

$$c_{il} = \begin{cases} 1 & \text{if task } \tau_i \text{ executes with frequency } f_l, \\ 0 & \text{otherwise.} \end{cases}$$

2) Continuous variable

$o_i$ optional cycles of task $\tau_i$.

For tractability reasons, $o_i$ is considered as continuous variable in the problem formulation. When the problem is solved, the result is rounded down. As the tasks execute typically hundreds of thousands of cycles, this impact is negligible [9].

Let $\mathcal{N} \triangleq \{1, \ldots, N\}$, $\mathcal{M} \triangleq \{1, \ldots, M\}$ and $\mathcal{L} \triangleq \{1, \ldots, L\}$. Since 1) each task can be assigned only one frequency level, and 2) each task is executed at only one processor, the binary variables $c_{il}$ and $q_{ik}$ must satisfy the following constraints:

$$\sum\nolimits_{l=1}^{L} c_{il} = 1, \ \forall i \in \mathcal{N}, \tag{2}$$

$$\sum\nolimits_{k=1}^{M} q_{ik} = 1, \ \forall i \in \mathcal{N}. \tag{3}$$

For the real-time constraints, since 1) the execution time of task $M_i + o_i$ cannot exceed the relative deadline $D_i$, and 2)

the tasks assigned to processor $\theta_k$ must be executed within the scheduling horizon $H$, we have

$$\sum\nolimits_{l=1}^{L} c_{il}\Big(w_{il}+\frac{o_i}{f_l}\Big) \le D_i,\ \forall i\in\mathcal{N}, \tag{4}$$

$$\sum\nolimits_{i=1}^{N} q_{ik}\Big[\sum\nolimits_{l=1}^{L} c_{il}\Big(w_{il}+\frac{o_i}{f_l}\Big)\Big] \le H,\ \forall k\in\mathcal{M}, \tag{5}$$

where $w_{il}=\frac{M_i}{f_l}$ is the execution time of the mandatory subtask $M_i$ with frequency $f_l$. $D_i$ is determined by the applications and $D_i \ge \min_{\forall l\in\mathcal{L}}\left\{\frac{M_i}{f_l}\right\}$.

Based on the energy model (1), the total energy consumed by $M$ processors within the horizon $H$ is bounded by

$$\sum\nolimits_{i=1}^{N}\sum\nolimits_{l=1}^{L} c_{il}\Big(w_{il}+\frac{o_i}{f_l}\Big)\left(P_l^s+P_l^d\right) + \Big[MH-\sum\nolimits_{i=1}^{N}\sum\nolimits_{l=1}^{L} c_{il}\Big(w_{il}+\frac{o_i}{f_l}\Big)\Big]P_0^s \le E_s, \tag{6}$$

where $MH-\sum_{i=1}^{N}\sum_{l=1}^{L} c_{il}\big(w_{il}+\frac{o_i}{f_l}\big)$ is the time of $M$ processors in the idle state during the scheduling horizon $H$ and $P_0^s$ is the static power of idle state.

It has been shown in [4], [15], [17], [28] that applications exist, e.g., Tier-2 coder in M-JPEG2000, where the QoS highly depends on its optional subtask and it can be represented as a linear function of the optional cycles. The linear function increases its value uniformly with the optional cycles. To quantify the relationship between QoS and optional cycles, each task $\tau_i$ is associated with a linear function $g_i(o_i)$. In order to maximize QoS function $\sum_{i=1}^{N} g_i(o_i)$ (or to minimize its negative), based on all the aforementioned constraints, the Primal Problem (PP) is formulated as

$$\textbf{PP}: \min_{\boldsymbol{c},\boldsymbol{q},\boldsymbol{o}}\ \Phi=-\sum\nolimits_{i=1}^{N} g_i(o_i)$$
$$\text{s.t.}\ \begin{cases}(2),(3),(4),(5),(6),\\ c_{il},q_{ik}\in\{0,1\},\ 0\le o_i\le O_i,\\ \qquad\qquad \forall i\in\mathcal{N},\ \forall l\in\mathcal{L},\ \forall k\in\mathcal{M}.\end{cases}$$

Due to the products of optimization variables $c_{il}o_i$ and $q_{ik}c_{il}o_i$ in the constraints (4), (5) and (6), the PP is an MINLP problem, which is difficult to directly solve because of the nonlinear items [12].

*Theorem 2.1:* The MINLP-based QoS-aware IC-tasks mapping problem, i.e., the PP, is $\mathcal{NP}$–hard.

*Proof:* Due to page limitations, the proof is excluded from the manuscript. However, the details can be found in [29]. ■

## III. Optimal Approach

In this section, we present our optimal approach for solving the problem defined in Section II.

### A. MILP formulation

We adopt the idea of *variable replacement* to efficiently solve the PP. In this way, the MINLP can be safely transformed to an MILP with a simpler structure, and, thus, easier to solve. To achieve this, we first introduce an intermediate (continuous) variable

$$t_i=\sum\nolimits_{l=1}^{L} c_{il}\Big(w_{il}+\frac{o_i}{f_l}\Big),\ \forall i\in\mathcal{N}, \tag{7}$$

which is the execution time of task $\tau_i$. Then, we linearize the nonlinear items $c_{il}o_i$ and $q_{ik}t_i$ according to *Lemma 3.1*.

*Lemma 3.1:* Given constants $s_1>0$ and $s_2>0$ and two constraint spaces $S_1=\{[h,b,x]|h=bx,-s_1\le x\le s_2,b\in\{0,1\}\}$ and $S_2=\{[h,b,x]|-bs_1\le h\le bs_2,h+bs_1-x-s_1\le 0,h-bs_2-x+s_2\ge 0,b\in\{0,1\}\}$, then $S_1\leftrightarrows S_2$.

*Proof:* Based on $h=bx$ and $-s_1\le x\le s_2$, we have $-bs_1\le h\le bs_2$. And further, we obtain $(b-1)(x+s_1)\le 0$ and $(b-1)(x-s_2)\ge 0$ due to $-s_1\le x\le s_2$ and $b\in\{0,1\}$. Hence, we have $h+bs_1-x-s_1\le 0$ and $h-bs_2-x+s_2\ge 0$. $S_1\Rightarrow S_2$ holds.

If $b=0$, based on $-bs_1\le h\le bs_2$, $h+bs_1-x-s_1\le 0$ and $h-bs_2-x+s_2\ge 0$, we have $h=0$ and $-s_1\le x\le s_2$. Similarly, if $b=1$, we have $-s_1\le h=x\le s_2$. $S_2\Rightarrow S_1$ holds. ■

Although the lower bound of $t_i$ is $\min_{\forall l\in\mathcal{L}}\left\{\frac{M_i}{f_l}\right\}\ge 0$ ($=0$ when $M_i=0$), we assume that $0\le t_i\le D_i$ since the range $\left[0,\min_{\forall l\in\mathcal{L}}\left\{\frac{M_i}{f_l}\right\}\right]$ is excluded by (7). Note that $c_{il}\in\{0,1\}$, $q_{ik}\in\{0,1\}$, $0\le o_i\le O_i$ and $0\le t_i\le D_i$. We introduce two intermediate (continuous) variables $h_{il}$ and $d_{ik}$ to replace the nonlinear items $c_{il}o_i$ and $q_{ik}t_i$, respectively. Therefore, the constraints (4) and (6) are safely replaced by

$$\sum\nolimits_{l=1}^{L}\Big(c_{il}w_{il}+\frac{h_{il}}{f_l}\Big)\le D_i,\ \forall i\in\mathcal{N}, \tag{8}$$

$$\sum\nolimits_{i=1}^{N}\sum\nolimits_{l=1}^{L}\Big(c_{il}w_{il}+\frac{h_{il}}{f_l}\Big)\left(P_l^d+P_l^s-P_0^s\right) + MHP_0^s\le E_s, \tag{9}$$

$$h_{il}\le c_{il}O_i,\ \forall i\in\mathcal{N},\ \forall l\in\mathcal{L}, \tag{10}$$

$$h_{il}\le o_i,\ \forall i\in\mathcal{N},\ \forall l\in\mathcal{L}, \tag{11}$$

$$o_i-h_{il}-(1-c_{il})\,O_i\le 0,\ \forall i\in\mathcal{N},\ \forall l\in\mathcal{L}. \tag{12}$$

Based on the intermediate variables $t_i$ and $h_{il}$, the constraint (5) is safely replaced by

$$t_i=\sum\nolimits_{l=1}^{L}\Big(c_{il}w_{il}+\frac{h_{il}}{f_l}\Big),\ \forall i\in\mathcal{N}, \tag{13}$$

$$\sum\nolimits_{i=1}^{N} d_{ik}\le H,\ \forall k\in\mathcal{M}, \tag{14}$$

$$d_{ik}\le q_{ik}D_i,\ \forall i\in\mathcal{N},\ \forall k\in\mathcal{M}, \tag{15}$$

$$d_{ik}\le t_i,\ \forall i\in\mathcal{N},\ \forall k\in\mathcal{M}, \tag{16}$$

$$t_i-d_{ik}-(1-q_{ik})\,D_i\le 0,\ \forall i\in\mathcal{N},\ \forall k\in\mathcal{M}, \tag{17}$$

where (10)–(12) and (15)–(17) are the additional constraints introduced by the linearization.

Consequently, the PP is rewritten as the MILP problem:

$$\textbf{PP1}: \min_{\substack{\boldsymbol{c},\boldsymbol{q},\boldsymbol{o},\\ \boldsymbol{t},\boldsymbol{h},\boldsymbol{d}}}\ \Phi=-\sum\nolimits_{i=1}^{N} g_i(o_i)$$
$$\text{s.t.}\ \begin{cases}(2),(3),(8)-(17),\\ c_{il},q_{ik}\in\{0,1\},\ 0\le o_i,h_{il}\le O_i,\\ \min_{\forall l\in\mathcal{L}}\left\{\frac{M_i}{f_l}\right\}\le t_i\le D_i,\ 0\le d_{ik}\le D_i,\\ \qquad\qquad \forall i\in\mathcal{N},\ \forall l\in\mathcal{L},\ \forall k\in\mathcal{M}.\end{cases}$$

*Remark 3.1: Lemma 3.1* implies that the variable replacement does not change the feasible region of the problem ($S_1\leftrightarrows S_2$). Since the objective functions of PP and PP1 are the same, solving PP1 is equivalent to solving PP, i.e., the optimal objective function values of PP1 and PP are the same.

### B. Optimal Joint DVFS Task Mapping

We propose an optimal algorithm, i.e., OJTM, to solve PP1 based on its special characteristics, i.e., the binary and continuous variables are coupled with each other linearly. The key idea of OJTM comes from Benders decomposition. Instead of considering all the variables and the constraints of PP1 simultaneously, OJTM decomposes the PP1 into a *Master Problem* (MP) and a *Slave Problem* (SP), which are solved iteratively through a feedback loop. By doing so, the computational complexity of the solution is significantly reduced [23], [30], even if the PP1 is non-convex. The structure of OJTM is shown in Fig. 1. The MP accounts for all the binary variables (task-to-processor allocation and frequency-to-task assignment) and the corresponding part of the objective function and the constraints of PP1. It also includes a set of constraints called *Benders cuts*. The SP includes all the continuous variables (task adjustment) and the associated constraints of PP1. The SP solution provides additional constraint, which is incorporated in the MP through the Benders cuts.

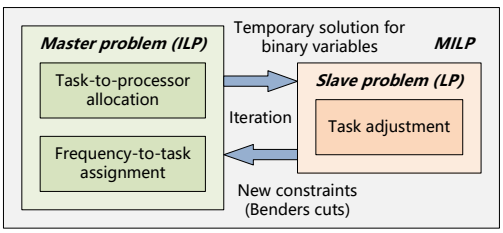



Fig. 1. The structure of OJTM algorithm.

For presentation convenience, matrices and vectors are used to denote the constraints and variables. Hence, the PP1 is reformulated as

$$\textbf{PP1}: \min_{\boldsymbol{x},\boldsymbol{y}} \ \boldsymbol{f}'\boldsymbol{y} \quad \text{s.t.} \ \begin{cases} \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1, \\ \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2, \end{cases}$$

where $\boldsymbol{x}$ is a binary vector, and $\boldsymbol{y}$ is a continuous vector. Vector $\boldsymbol{f}$ represents the objective function coefficients, and $\boldsymbol{f}'$ represents the transposition of $\boldsymbol{f}$. Matrices $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ represent the constraints' coefficients. $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are the $u$-dimensional and $v$-dimensional vectors, respectively.

Based on the structure of the PP1, at the $m^{th}$ iteration, the MP is formulated as

$$\textbf{MP}: \Phi_l(m) = \min_{\boldsymbol{x},\hat{\Phi}} \ \hat{\Phi} \quad \text{s.t.} \ \begin{cases} \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1, \\ \hat{\Phi} \geq \boldsymbol{\lambda}(\varsigma)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2), \ \forall \varsigma \in \mathcal{A}, \\ 0 \geq \hat{\boldsymbol{\lambda}}(\vartheta)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2), \ \forall \vartheta \in \mathcal{B}, \end{cases} \tag{18}$$

where $\hat{\Phi} \geq \boldsymbol{\lambda}(\varsigma)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ $(\forall \varsigma \in \mathcal{A})$ and $0 \geq \hat{\boldsymbol{\lambda}}(\vartheta)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ $(\forall \vartheta \in \mathcal{B})$ are the sets of feasibility and infeasibility constraints (Benders cuts), respectively. $\boldsymbol{\lambda}(m)$ and $\hat{\boldsymbol{\lambda}}(m)$ are the solutions of the Dual Slave Problem (DSP) (19) and the Dual Feasibility Check Problem (DFCP) (21) at the $m^{th}$ iteration, respectively. $\mathcal{A}$ and $\mathcal{B}$ are the sets of iterations that the DSP has bounded and unbounded solutions, respectively. Note that 1) the objective function of the PP1 only contains continuous variables $\boldsymbol{y}$, and 2) the MP only considers binary variables $\boldsymbol{x}$. We introduce an auxiliary (continuous) variable $\hat{\Phi}$ for the MP as the objective function, where $\hat{\Phi}$ has the same physical meaning as the objective function of the PP1. The relationship between $\hat{\Phi}$ and $\Phi$ is explained in the next section.

The corresponding SP is formulated as

$$\textbf{SP}: \Phi_u(m) = \min_{\boldsymbol{y} \succeq 0} \ \boldsymbol{f}'\boldsymbol{y} \quad \text{s.t.} \ \boldsymbol{C}\boldsymbol{x}(m) + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2,$$

where $\boldsymbol{x}(m)$ is the MP solution at the $m^{th}$ iteration. In fact, the SP is "identical" to PP1: the formulations are the same, except that the binary variables $\boldsymbol{x}(m)$ in the SP are fixed.

Denote $\Phi^*$ as the optimal objective function value of the PP1. Note that the MP (18) only contains the frequency-to-task and task-to-processor information, i.e., $\boldsymbol{x}(m)$. Compared with the PP1, the task adjustment related constraints are relaxed. Solving the MP yields a *lower bound* of $\Phi^*$, i.e., $\Phi_l(m)$. On the other hand, $\boldsymbol{x}(m)$ may be just a feasible solution (not optimal yet), and SP is a minimization problem. Solving the SP with $\boldsymbol{x}(m)$ yields an *upper bound* of $\Phi^*$, i.e., $\Phi_u(m)$ (the proof is provided in *Theorem 3.1*). Therefore, we have $\Phi_l(m) \leq \Phi^* \leq \Phi_u(m)$. At each iteration, introducing a new feasibility constraint (20) or infeasibility constraint (22) into the MP, the gap between the upper and lower bounds, i.e., $\Phi_u(m) - \Phi_l(m)$, gradually reduces (the proof is provided in *Lemma 3.3*), and the iterations stop when $\Phi_u(m) - \Phi_l(m) \leq \varepsilon$.

*Definition 3.1 ($\varepsilon$-optimal solution):* A solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ of the PP1 is an $\varepsilon$-optimal solution, if it satisfies the constraint $0 \leq \boldsymbol{f}'\tilde{\boldsymbol{y}} - \Phi^* \leq \varepsilon$.

*Lemma 3.2:* Based on the iteration termination criterion $\Phi_u(m) - \Phi_l(m) \leq \varepsilon$, the solution $(\boldsymbol{x}(m), \boldsymbol{y}(m))$ obtained by OJTM is an $\varepsilon$-optimal solution.

*Proof:* Substituting the solution $(\boldsymbol{x}(m), \boldsymbol{y}(m))$ into the PP1, we have $\Phi_l(m) \leq \boldsymbol{f}'\boldsymbol{y}(m) = \Phi_u(m)$. Since $\Phi_u(m) - \Phi_l(m) \leq \varepsilon$, we get $0 \leq \boldsymbol{f}'\boldsymbol{y}(m) - \Phi^* \leq \varepsilon$, which means $(\boldsymbol{x}(m), \boldsymbol{y}(m))$ is an $\varepsilon$-optimal solution. ■

If $\varepsilon$ is a small positive value, the optimal value $\Phi^*$ is found by OJTM (the proof is provided in *Theorem 3.2*). The iteration process is summarized as follows:

*1) Step 1 – Initialization:* Initialize the iteration counter $m = 0$, the solution $\boldsymbol{x}(0)$ of the MP, the lower bound $\Phi_l(0) = -\infty$, and the upper bound $\Phi_u(0) = +\infty$. The feasibility and infeasibility constraints are set to null. Note that the initial solution $\boldsymbol{x}(0)$ can be given arbitrarily, as long as it satisfies the constraints (2) and (3).

*2) Step 2 – Solving SP:* Assume that the current iteration is $m$. Since the SP is a LP problem, the optimal objective function values of the SP and its dual problem are equivalent due to the strong duality [31]. Introducing the positive Lagrange multipliers $\boldsymbol{\lambda} \triangleq [\lambda_i]$ $(1 \leq i \leq v)$ to the SP, the associated dual problem (DSP) is

$$\textbf{DSP}: \max_{\boldsymbol{\lambda} \succeq 0} \ \boldsymbol{\lambda}'\left(\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2\right) \quad \text{s.t.} \ \boldsymbol{f} + \boldsymbol{D}'\boldsymbol{\lambda} \succeq 0. \tag{19}$$

Since the DSP is a LP problem, the solution can be found quickly using standard algorithms, such as simplex method or interior point method [9].

*3) Step 3 – Solving MP:* Depends on the following cases:

1) The DSP is *infeasible*: The SP has an unbounded solution. Hence, the PP1 has no feasible solution.
2) The DSP has a *bounded solution* $\boldsymbol{\lambda}(m)$: The SP is feasible due to the strong duality, and $\mathcal{A} \leftarrow \{m\} \cup \mathcal{A}$. Assume that $\hat{\Phi}(m)$ and $\boldsymbol{y}(m)$ are the solution of the MP and the SP at the $m^{th}$ iteration, respectively. Since i) $\boldsymbol{x}(m)$ is a feasible solution (not optimal) of PP1, and ii) $\boldsymbol{f}'\boldsymbol{y}(m) = \boldsymbol{\lambda}(m)'(\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2)$ due to the strong duality, the upper bound of $\Phi^*$ at the $m^{th}$ iteration is updated by $\Phi_u(m) = \min\{\Phi_u(m-1), \boldsymbol{\lambda}(m)'(\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2)\}$. Note that i) $\hat{\Phi}(m) < \boldsymbol{\lambda}(m)'(\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2)$, and ii) $\boldsymbol{x}(m)$ is not the optimal solution. In order to avoid selecting the non-optimal solution $\boldsymbol{x}(m)$ again, a new feasibility constraint
$$\hat{\Phi} \geq \boldsymbol{\lambda}(m)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2) \tag{20}$$
is generated and added into the MP at the $(m+1)^{th}$ iteration.
3) The DSP has an *unbounded solution*, i.e., $\boldsymbol{\lambda}(m)'(\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2) = +\infty$: Due to the strong duality, the SP has no feasible solution under given solution $\boldsymbol{x}(m)$, and $\mathcal{B} \leftarrow \{m\} \cup \mathcal{B}$. For the SP, its feasibility is related to the constraints rather than the objective function. This problem may feasible if the positive variables $\boldsymbol{\xi} \triangleq [\xi_i]$ $(1 \leq i \leq v)$ are introduced to relax the constraints. Based on this idea, we construct a Feasibility Check Problem (FCP)
$$\begin{aligned} \textbf{FCP}: \min_{\boldsymbol{\xi} \succeq 0} \ & \boldsymbol{1}'\boldsymbol{\xi} \\ \text{s.t.} \ & \boldsymbol{C}\boldsymbol{x}(m) + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2 + \boldsymbol{\xi}. \end{aligned}$$
Since the FCP is a LP problem, the strong duality exists between the FCP and its dual problem. Introducing the positive Lagrange multipliers $\hat{\boldsymbol{\lambda}} = \left[\hat{\lambda}_i\right]$ $(1 \leq i \leq v)$ to the FCP, the associated dual problem (DFCP) is
$$\begin{aligned} \textbf{DFCP}: \max_{\hat{\boldsymbol{\lambda}} \succeq 0} \ & \hat{\boldsymbol{\lambda}}'(\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2) \\ \text{s.t.} \ & \boldsymbol{1} - \hat{\boldsymbol{\lambda}} \succeq 0. \end{aligned} \tag{21}$$
The presented method used to solve DSP can be also applied here to solve FCP and DFCP, as both are LP problems. Assume that $\boldsymbol{\xi}(m)$ and $\hat{\boldsymbol{\lambda}}(m)$ are the solutions of FCP and DFCP, respectively. If the SP exists infeasible constraints, the related relax variables are non-zero, while the others are zero, i.e., $\boldsymbol{1}'\boldsymbol{\xi}(m) > 0$. Due to the strong duality, we have $\boldsymbol{1}'\boldsymbol{\xi}(m) = \hat{\boldsymbol{\lambda}}(m)'(\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2) > 0$. In order to avoid selecting the infeasible solution $\boldsymbol{x}(m)$ again, a new infeasibility constraint
$$0 \geq \hat{\boldsymbol{\lambda}}(m)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2) \tag{22}$$
is generated and added into the MP at the $(m+1)^{th}$ iteration.

Note that in (20) and (22), all the parameters are constant except $\hat{\Phi}$ and $\boldsymbol{x}$. These are the variables of the MP at the $(m+1)^{th}$ iteration. With this new added feasibility/infeasibility constraint, the MP is solved again to obtain a new solution $\boldsymbol{x}(m+1)$ at the next iteration.

*Remark 3.2:* Note that 1) the SP has the same objective function as the PP1, i.e., $\boldsymbol{f}'\boldsymbol{y}$, and 2) the SP and the DSP are equivalent due to the strong duality, i.e., they have the same optimal objective function values. From (20), we can see that the auxiliary variable $\hat{\Phi}$ has the same physical meaning as the objective function of PP1.

*Remark 3.3:* We solve the DSP rather than solving the SP directly, because 1) the SP and its dual problem are equivalent due to the strong duality, and 2) we can construct feasibility and infeasibility constraints through the solution of the DSP. The reason why we select the solution of the DFCB rather than the solution of the FCB to construct infeasibility constraint is that $\hat{\boldsymbol{\lambda}}'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ is a function with respect to $\boldsymbol{x}$ but not $\boldsymbol{1}'\boldsymbol{\xi}$, i.e., $0 \geq \boldsymbol{1}'\boldsymbol{\xi}$ is an invalid constraint for the MP.

### C. Convergence analysis

Equation (18) shows that 1) the MP is composed of the binary variables $\boldsymbol{x}$ and a continuous variable $\hat{\Phi}$, 2) the real constraints are $\boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1$ and $0 \geq \hat{\boldsymbol{\lambda}}(\vartheta)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ $(\forall \vartheta \in \mathcal{B})$, and 3) $\boldsymbol{\lambda}(\varsigma)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ $(\forall \varsigma \in \mathcal{A})$ can be treated as objective functions. Therefore, the MP can be solved by only considering the binary variables $\boldsymbol{x}$ [30]. Comparing the MP with the following optimization problem, which has the same constraints, we have $\hat{\Phi}(m) = \max_{\forall i \in \mathcal{A}} \left\{\hat{\Phi}_r(i)\right\}$.
$$\begin{aligned} \textbf{MP}_i: \hat{\Phi}_r(i) = & \min_{\boldsymbol{x}} \boldsymbol{\lambda}(i)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2) \\ & \text{s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1, \\ 0 \geq \hat{\boldsymbol{\lambda}}(\vartheta)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2), \ \forall \vartheta \in \mathcal{B}. \end{cases} \end{aligned}$$

*Theorem 3.1:* Solving the MP and the SP at the $m^{th}$ iteration, we obtain a lower bound $\Phi_l(m)$ and an upper bound $\Phi_u(m)$ of the optimal value of the objective function $\Phi^*$, respectively.

*Proof:* First, we prove that $\Phi_l(m) = \hat{\Phi}(m)$ is a lower bound of $\Phi^*$, where $\hat{\Phi}(m)$ is the solution of the MP at the $m^{th}$ iteration. Without loss of generality, we assume that $\hat{\Phi}(m) = \hat{\Phi}_r(\rho) = \max_{\forall i \in \mathcal{A}} \left\{\hat{\Phi}_r(i)\right\}$, where $\rho \in \mathcal{A}$. Hence, we have
$$\begin{aligned} \hat{\Phi}(m) &= \min_{\boldsymbol{x}} \boldsymbol{\lambda}(\rho)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2) \\ &\leq \boldsymbol{\lambda}(\rho)'(\boldsymbol{C}\boldsymbol{x}^* - \boldsymbol{b}_2) \qquad (23a) \\ &\leq \max_{\boldsymbol{\lambda} \succeq 0} \boldsymbol{\lambda}'(\boldsymbol{C}\boldsymbol{x}^* - \boldsymbol{b}_2) = \Phi^*, \qquad (23b) \end{aligned}$$
where $\min_{\boldsymbol{x}} \boldsymbol{\lambda}(\rho)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ is the objective function value of the $\text{MP}_\rho$, and $\max_{\boldsymbol{\lambda} \succeq 0} \boldsymbol{\lambda}'(\boldsymbol{C}\boldsymbol{x}^* - \boldsymbol{b}_2)$ is the objective function value of the DSP. (23a) holds since $\boldsymbol{x}^*$ is not the optimal solution of problem $\min_{\boldsymbol{x}} \boldsymbol{\lambda}(\rho)'(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$. (23b) holds since solving the DSP with the optimal binary variables $\boldsymbol{x}^*$ we find optimal value $\Phi^*$. (23) shows that $\Phi_l(m) = \hat{\Phi}(m)$ is a lower bound of $\Phi^*$.

In the following, we prove that $\Phi_u(m) = \min\{\Phi_u(m-1), \boldsymbol{\lambda}(m)'(\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2)\}$ is an upper bound of $\Phi^*$. Note that we have $\min\{\Phi_u(m-1), \boldsymbol{\lambda}(m)'(\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2)\} = \min_{1 \leq i \leq m}\{\boldsymbol{\lambda}(i)'(\boldsymbol{C}\boldsymbol{x}(i) - \boldsymbol{b}_2)\}$, where $\boldsymbol{x}(i)$ and $\boldsymbol{\lambda}(i)$ are the solutions of the MP and the DSP at the $i^{th}$ iteration, respectively. Depending on the solution of the DSP, the objective function value of the DSP, i.e., $\boldsymbol{\lambda}(i)'(\boldsymbol{C}\boldsymbol{x}(i) - \boldsymbol{b}_2)$, can be either finite or infinite. If the DSP has an unbounded solution, i.e., $\boldsymbol{\lambda}(i)'(\boldsymbol{C}\boldsymbol{x}(i) - \boldsymbol{b}_2) = +\infty$, it is obvious that $+\infty$ is an upper bound of $\Phi^*$. Thus, we consider the case when the DSP has a bounded solution $\boldsymbol{\lambda}(i)$. Due to the strong

SP and DSP duality, we have

$$\boldsymbol{\lambda}(i)'(\boldsymbol{C}\boldsymbol{x}(i)-\boldsymbol{b}_2)=\min_{\boldsymbol{y}\succeq 0}\boldsymbol{f}'\boldsymbol{y}|\boldsymbol{x}(i)$$
$$\geq \min_{\boldsymbol{y}\succeq 0}\boldsymbol{f}'\boldsymbol{y}|\boldsymbol{x}^*=\Phi^*,\ 1\leq i\leq m,$$

where $\min_{\boldsymbol{y}\succeq 0}\boldsymbol{f}'\boldsymbol{y}|\boldsymbol{x}(i)$ is the objective function value of the DSP with $\boldsymbol{x}(i)$. Thus, $\Phi_u(m)=\min_{1\leq i\leq m}\{\boldsymbol{\lambda}(i)'(\boldsymbol{C}\boldsymbol{x}(i)-\boldsymbol{b}_2)\}$ is an upper bound of $\Phi^*$. ■

*Lemma 3.3:* The lower bound sequence $\{\Phi_l(m)\}$ is increasing, while the upper bound sequence $\{\Phi_u(m)\}$ is decreasing.

*Proof:* Assume that $(\hat{\Phi}(m),\boldsymbol{x}(m))$ is the solution of the MP at the $m^{th}$ iteration. If $\boldsymbol{x}(m)$ are the non-optimal (infeasible) values of variables $\boldsymbol{x}$, $\boldsymbol{x}(m)$ are excluded by adding new constraint $\hat{\Phi}\geq\boldsymbol{\lambda}(m)'(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)$ $(0\geq\hat{\boldsymbol{\lambda}}(m)'(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2))$ into the MP at the $(m+1)^{th}$ iteration. Note that the variables of the MP, i.e., $\boldsymbol{x}$ and $\hat{\Phi}$, are coupled with each other linearly. If $\boldsymbol{x}(m)$ are excluded, $\hat{\Phi}(m)$ is excluded as well. Since 1) the aim of the MP is to minimize the objective function, 2) the non-optimal values of $\Phi^*$, i.e., $\{\hat{\Phi}(0),\ldots,\hat{\Phi}(m)\}$, have been excluded by the constraints $\hat{\Phi}\geq\boldsymbol{\lambda}(\varsigma)'(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)$ $(\forall\varsigma\in\mathcal{A})$ and $0\geq\hat{\boldsymbol{\lambda}}(\vartheta)'(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)$ $(\forall\vartheta\in\mathcal{B})$, and 3) with iteration number $m$ increasing, more constrains are added into the MP (i.e., the feasible region of the MP will shrink), $\Phi_l(m+1)=\hat{\Phi}(m+1)$ is larger than the previous lower bounds $\{\Phi_l(0),\ldots,\Phi_l(m)\}$.

Assume that $\Phi_u(m+1)>\Phi_u(m)$. This contradicts the fact that $\Phi_u(m+1)\leq\Phi_u(m)$ and $\Phi_u(m+1)\leq\boldsymbol{\lambda}(m+1)'(\boldsymbol{C}\boldsymbol{x}(m+1)-\boldsymbol{b}_2)$ due to $\Phi_u(m+1)=\min\{\Phi_u(m),\boldsymbol{\lambda}(m+1)'(\boldsymbol{C}\boldsymbol{x}(m+1)-\boldsymbol{b}_2)\}$. Therefore, $\Phi_u(m+1)$ is no larger than the previous upper bounds $\{\Phi_u(0),\ldots,\Phi_u(m)\}$. ■

*Theorem 3.2:* At each iteration with a new feasibility constraint (20) or infeasibility constraint (22) added into the MP, the solution obtained by OJTM converges to the global optimal value within a finite number of iterations.

*Proof:* At each iteration, there is a new constraint ((20) or (22)) added into the MP to exclude the non-optimal or the infeasible values of the binary variables $\boldsymbol{x}$. Note that 1) the lower bound sequence $\{\Phi_l(0),\ldots,\Phi_l(m)\}$ is increasing, 2) the upper bound sequence $\{\Phi_u(0),\ldots,\Phi_u(m)\}$ is decreasing, and 3) the dimension of binary variables $\boldsymbol{x}$ is finite. According to the *Theorem 2.4.* in [32], the solution converges to the global optimal value within a finite number of iterations. ■

## IV. Heuristic Approach

The computational complexity of OJTM is dominated by the cost of solving the ILP-based MP at each iteration. Although the solution provided by OJTM is optimal, this method cannot be used to efficiently solve large problem sizes. To enhance the scalability of the proposed approach, we provide a novel heuristic algorithm, i.e., HJTM, to efficiently solve the PP. The reasons why we target PP rather than PP1 are that 1) these two problems are equivalent, and 2) the PP contains less variables and constraints making the heuristic less complex. The main idea of the HJTM is that the most important step of solving the PP is to find proper task-to-processor and frequency-to-task decisions (similar to basic idea of the OJTM). If the binary variables $\boldsymbol{c}$ and $\boldsymbol{q}$ are determined, the PP reduces to a LP problem. The structure of HJTM is shown in Fig. 2. HJTM has three steps. During the first step, we obtain feasible frequency-to-task assignment. Having determined the frequency-to-task decision, the problem reduces to finding feasible task allocation to the processors. In the final step, the optional cycles are defined to meet the real-time and the energy supply constraints.

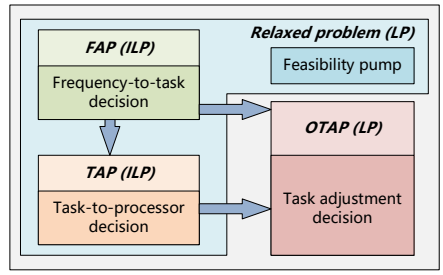



Fig. 2. The structure of HJTM algorithm.

### A. Frequency-to-task Assignment

Since the mandatory subtasks must be always executed, we initially consider only the frequency assignment of the mandatory subtasks and our aim is to minimize the energy consumption by assigning proper frequency to each task. From the constraint (6), we observe that if $o_i=0$ $(\forall i\in\mathcal{N})$, the consumed energy of $M$ processors reduces to $\sum_{i=1}^{N}\sum_{l=1}^{L}c_{il}w_{il}\big(P_l^s+P_l^d-P_0^s\big)+MHP_0^s$. Since item $MHP_0^s$ is fixed, the Frequency Assignment Problem (FAP) is formulated as

$$\textbf{FAP}:\min_{\boldsymbol{c}}\sum\nolimits_{i=1}^{N}\sum\nolimits_{l=1}^{L}c_{il}w_{il}\left(P_l^s+P_l^d-P_0^s\right)$$
$$\text{s.t.}\ \begin{cases}(2),\\ \sum\nolimits_{l=1}^{L}c_{il}w_{il}\leq D_i,\ \forall i\in\mathcal{N}.\end{cases}$$

### B. Task-to-processor Allocation

Solving the FAP, we obtain the execution time of task $\tau_i$'s mandatory subtask, i.e., $\sum_{l=1}^{L}c_{il}\frac{M_i}{f_l}$. Taking task allocation decision $\boldsymbol{q}$ into account, the execution time of processor $\theta_k$ is computed by $\Lambda(k)=\sum_{i=1}^{N}q_{ik}\big(\sum_{l=1}^{L}c_{il}\frac{M_i}{f_l}\big)$. Since the processors have the same voltage/frequency levels $\{(v_1,f_1),\ldots,(v_L,f_L)\}$, different task allocation schemes will not influence the solution of FAP, i.e., the task-to-processor decision can be made when frequency-to-task assignment is done. We introduce an auxiliary(continuous) variable $U$, which is denoted as the maximum task execution time among the processors, i.e., $U=\max_{\forall k\in\mathcal{M}}\{\Lambda(k)\}$, and our aim is to minimize $U$ by allocating the selected tasks to each processor. Thus, the Task Allocation Problem (TAP) is formulated as

$$\textbf{TAP}:\min_{\boldsymbol{q},U}\ U$$
$$\text{s.t.}\ \begin{cases}U\geq\sum\nolimits_{i=1}^{N}q_{ik}\Big(\sum\nolimits_{l=1}^{L}c_{il}\dfrac{M_i}{f_l}\Big),\ \forall k\in\mathcal{M},\\ (3),\\ \sum\nolimits_{i=1}^{N}q_{ik}\Big(\sum\nolimits_{l=1}^{L}c_{il}\dfrac{M_i}{f_l}\Big)\leq H,\ \forall k\in\mathcal{M}.\end{cases}$$

Note that the TAP is composed of binary variables $\boldsymbol{q}$ and a continuous variable $U$. It can be solved using a similar method as the one presented for the solution of the MP.

### C. Optional Task Adjustment

Based on frequency-to-task assignment decision $\boldsymbol{c}$ and task-to-processor allocation decision $\boldsymbol{q}$, the PP reduces to the following Optional Task Adjustment Problem (OTAP):

$$\textbf{OTAP}: \min_{\boldsymbol{o}} \quad -\sum_{i=1}^{N} g_i(o_i)$$
$$\text{s.t.} \quad (4), (5), (6) \text{ under given c and q.}$$

Although the FAP and the TAP are ILP, compared with LP, they are still complex to solve directly, especially when problem size is large. To reduce the computational complexity, we relax FAP and TAP to LP problems. In particular, for the FAP (TAP), we introduce continuous variables $\hat{\boldsymbol{q}}$ ($\hat{\boldsymbol{c}}$), where $0 \leq \hat{q}_{ik} \leq 1$ ($0 \leq \hat{c}_{il} \leq 1$), to replace the original integer variables $\boldsymbol{q}$ ($\boldsymbol{c}$). And then, we solve the relaxed problem (LP) and round the solution $\hat{\boldsymbol{q}}$ ($\hat{\boldsymbol{c}}$) to the nearest binary matrix that is feasible to the FAP (TAP). Such a binary matrix can be efficiently found by using the Feasibility Pump (FP) method proposed in [33].

## V. Experimental Results

The values and the tuned parameters of the experimental set-up are summarized in Table II. The processor parameters are adopted from [10], where the platform is based on 70 nm technology and the accuracy of the parameters' values has been verified by SPICE simulations. The number of processors $M$ is tuned from 4 to 10 with a step of 2. The WCECs of tasks are assumed to be in the range provided in Table II. This range is calculated in [34] from the MiBench and MediaBench benchmark suites. The number of tasks $N$ is tuned from 10 to 50 with a step of 10. The QoS function of the tasks, i.e., the objective function of the PP, is adopted from [16], [17]. The scheduling horizon $H$ is assumed to be proportional to the average processor workload, and the energy supply $E_s$ is set to $E_s = \eta E_h$, where $E_h$ is the minimum energy required to execute $\{M_1 + O_1, \ldots, M_N + O_N\}$ cycles, $\eta \in [0, 1]$ is an energy efficiency factor, and $\eta$ is tuned from 0.8 to 0.9 with a step of 0.05. Note that different processor and task parameters lead to different values in the parameters $\{\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{f}, \boldsymbol{b}_1, \boldsymbol{b}_2\}$ for the PP1. However, the problem structures under different values of parameters are the same, and, thus, the proposed methods are still applicative.

TABLE II. Experimental set-up

| Processor $\theta_k$ characteristics | | | | | |
|---|---|---|---|---|---|
| $v_l$ (V) | 0.65 | 0.7 | 0.75 | 0.8 | 0.85 |
| $f_l$ (GHz) | 1.01 | 1.26 | 1.53 | 1.81 | 2.10 |
| $P_l^d$ (mW) | 184.9 | 266.7 | 370.4 | 498.9 | 655.5 |
| $P_l^s$ (mW) | 246 | 290.1 | 340.3 | 397.6 | 462.7 |
| $P_0^s$ ($\mu$W) | 80 | | | | |
| Task $\tau_i$ characteristics | | | | | |
| $M_i, O_i \in [4 \times 10^7, 6 \times 10^8]$ | | | $D_i = \min_{\forall l \in \mathcal{L}} \left\{ \frac{M_i + O_i}{f_l} \right\}$ | | |
| Objective function | | | | | |
| $\sum_{i=1}^{N} g_i(o_i) = \sum_{i=1}^{N} o_i$ | | | | | |
| Constraints | | | | | |
| $H = \left\lceil \frac{N}{M} \right\rceil \frac{\sum_{i=1}^{N} D_i}{N}$ | | | | | |
| $E_s = \eta E_h$ | | | | | |
| $E_h = MHP_0^s + \sum_{i=1}^{N} \left[ \min_{\forall l \in \mathcal{L}} \frac{M_i + O_i}{f_l} \left( P_l^s + P_l^d - P_0^s \right) \right]$ | | | | | |
| Tuned parameters | | | | | |
| | $M$ | $N$ | $\eta$ | | |
| Min/Max/Step | 4/10/2 | 10/50/10 | 0.8/0.9/0.05 | | |

We compare the behavior (QoS and computation time) of the proposed OJTM and HJTM with: 1) optimal approaches, i.e. Branch and Bound method (B&B) [22], which is known to provide optimal solution for the MILP problem – as far as we know no optimal algorithm exist for the problem formulation PP1, 2) stochastic approaches, i.e. Genetic Algorithm (GA) [35], and 3) heuristics, i.e. Adaptive Task Allocation (ATA) [16] – a two-step optimization approach. The simulations are performed on a laptop with quad-core 2.5 GHz Intel i7 processor and 16 GB RAM, and the algorithms are implemented in Matlab 2016a.

The QoS achieved by OJTM for all $M$, $N$ and $\eta$ parameters is shown in Fig. 3. During the simulations, the task sets in the experiments with $N = i$ ($i = 20, \ldots, 50$) and $N = i - 1$ are correlated, i.e., the task set in the experiment with $N = i$ is extended based on the task set in the experiment with $N = i - 1$. Note that 1) the scheduling horizon $H$ is proportional to the average processor workload, 2) the energy supply $E_s$ is proportional to the number of tasks $N$, and 3) the QoS is given by $\sum_{i=1}^{N} o_i$. The QoS generally increases with respect to 1) $N$ and 2) $\eta$ under given $M$ and $N$.

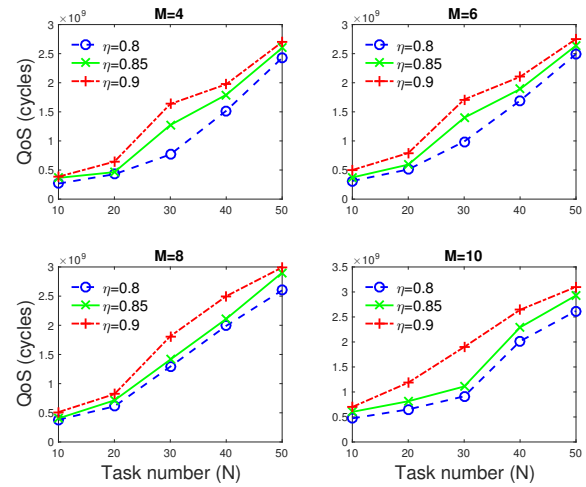



Fig. 3. QoS of OJTM with $M$, $N$ and $\eta$ varying.

We further explore the behavior of OJTM and HJTM by comparing the QoS gain of each method with the optimal B&B for all the aforementioned cases. Fig. 4 shows the statistical property of the QoS gain. The QoS gain of OJTM and HJTM (OJTM and B&B) is given by $\frac{Q_o(M,N,\eta) - Q_h(M,N,\eta)}{Q_o(M,N,\eta)}$ $\left(\frac{Q_o(M,N,\eta) - Q_b(M,N,\eta)}{Q_o(M,N,\eta)}\right)$, where $Q_b(M, N, \eta)$, $Q_o(M, N, \eta)$ and $Q_h(M, N, \eta)$ are the QoS achieved by B&B, OJTM and HJTM under given $M$, $N$ and $\eta$ parameters, respectively. The box plot "OJTM vs HJTM" ("OJTM vs B&B") with $\eta = 0.8$ shows the statistical property of data set $\left\{\frac{Q_o(M,N,0.8) - Q_h(M,N,0.8)}{Q_o(M,N,0.8)}\right\}$ $\left(\left\{\frac{Q_o(M,N,0.8) - Q_b(M,N,0.8)}{Q_o(M,N,0.8)}\right\}\right)$ for all $M$ and $N$ parameters. On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the $25^{th}$ and $75^{th}$ percentiles, respectively. The whiskers extend to the most extreme data points that are not considered outliers and the outliers are plotted individually using the '+' symbol. From Fig. 4, we observe that 1) the solutions given by B&B and OJTM are same, and, thus, OJTM also finds the optimal solution, and 2) OJTM achieves higher

QoS (26.3% in average) than that of HJTM.

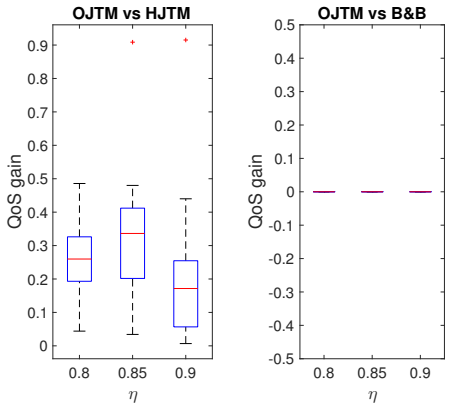
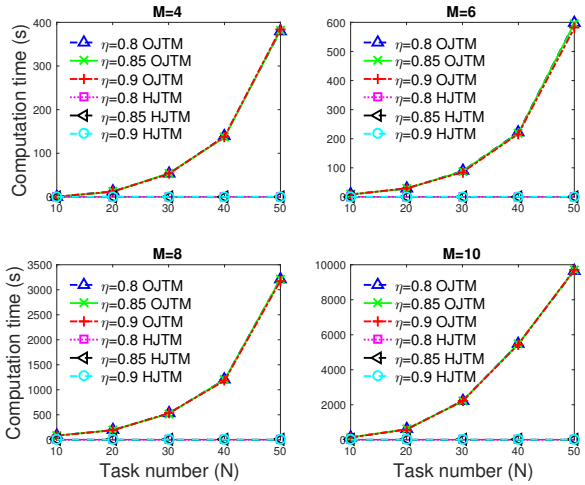



Fig. 4. QoS gain of OJTM, HJTM and B&B with $M$, $N$ and $\eta$ varying.

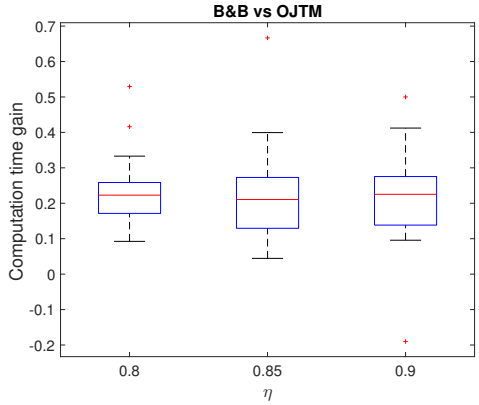
The computation time of OJTM, HJTM and B&B is evaluated in Fig. 5. From Fig. 5(a), which shows the OJTM (HJTM) computation time for solving the PP1 (PP), we observe that 1) the HJTM computation time is almost unchanged with $M$, $N$ and $\eta$ varying, and 2) when $M$ and $N$ are fixed and $\eta$ is tuned from 0.8 to 0.9, the changes of the OJTM computation time are small, compared with the changes of the OJTM computation time with $M$ and $N$ varying. Therefore, compared with OJTM, HJTM has a negligible computation time. We further compare the computation time gain of OJTM and B&B for all $M$, $N$ and $\eta$ parameters. The result is shown in Fig. 5(b). Denote $T_b(M,N,\eta)$ and $T_o(M,N,\eta)$ as the computation time of B&B and OJTM under given $M$, $N$ and $\eta$ parameters, respectively. The box plot "B&B vs OJTM" with $\eta = 0.8$ shows the statistical property of data set $\left\{\frac{T_b(M,N,0.8)-T_o(M,N,0.8)}{T_b(M,N,0.8)}\right\}$ for all $M$ and $N$ parameters. With $N$ increasing, the computation time of B&B and OJTM grows. However, OJTM takes a shorter computation time than that of B&B (22.6% in average). For an optimization problem, its computational complexity increases significantly with the number of variables and constraints. Solving iteratively smaller problems, i.e., the MP and the SP, is more efficient than solving a single large problem. This result agrees with the comparison of [36]: the decomposition-based method is faster than the B&B for lager problem instances.

We define the OJTM convergence iteration as the number of iterations to achieve $\Phi_u(m) - \Phi_l(m) \leq \varepsilon$, where $\varepsilon = 0.01$. Fig. 6 shows that with $M$ and $N$ increasing, more constraints are involved into the problem, and, thus, a higher number of iterations is required to converge to the optimal solution. We observe that when $M \leq 8$ and $N \leq 40$, the OJTM convergence iteration almost increases linearly with $M$ and $N$.

Denote $Q_g(M,N,\eta)$ and $T_g(M,N,\eta)$ as the QoS achieved by GA and the computation time of GA under given $M$, $N$ and $\eta$ parameters, respectively. The QoS and computation time gains of OJTM and GA, i.e., $\left\{\frac{Q_o(M,N,\eta)-Q_g(M,N,\eta)}{Q_o(M,N,\eta)}\right\}$ and $\left\{\frac{T_g(M,N,\eta)-T_o(M,N,\eta)}{T_g(M,N,\eta)}\right\}$, for all $M$, $N$ and $\eta$ parameters, are shown in Fig. 7. From this figure we observe that 1) OJTM achieves a higher QoS than that of GA (7.6% in average), and 2) OJTM takes a shorter computation time than that of GA (35.6% in average). Although GA can solve complex nonlinear programming problem (non-convex), such as MINLP,




(a) Computation time of OJTM and HJTM with $M$, $N$ and $\eta$ varying.




(b) Computation time gain of OJTM and B&B with $M$, $N$ and $\eta$ varying.

Fig. 5. Comparison of OJTM, HJTM and B&B computation time with $M$, $N$ and $\eta$ varying.

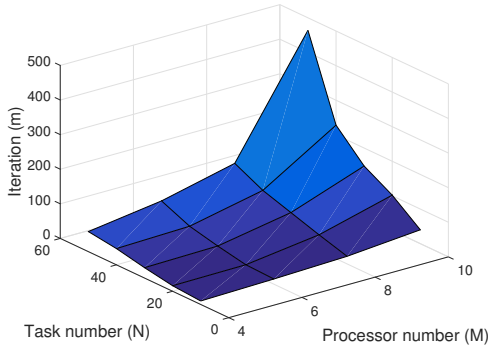



Fig. 6. OJTM convergence iteration.

the optimality of the solution is hard to guarantee. Compared with OJTM, the GA structure is more complex, as in each iteration GA needs to generate new populations. Therefore, the problem transformation from MINLP-based PP to MILP-based PP1 is necessary, since it can simplify the structure of the problem, and, thus, the optimal solution is much easier to

find.

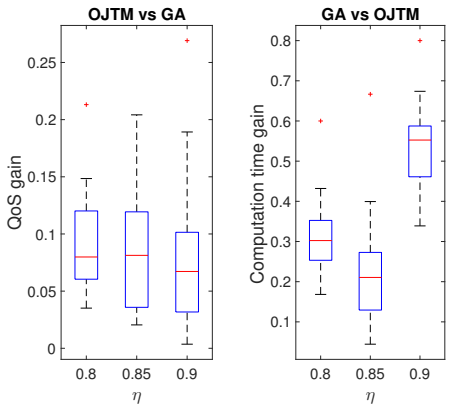



Fig. 7. QoS and computation time gains of OJTM and GA with $M$, $N$ and $\eta$ varying.

Denote $Q_a(M, N, \eta)$ and $T_a(M, N, \eta)$ as the QoS achieved by ATA and the computation time of ATA under given $M$, $N$ and $\eta$ parameters, respectively. The QoS gain of HJTM and ATA, i.e., $\{\frac{Q_h(M,N,\eta)-Q_a(M,N,\eta)}{Q_h(M,N,\eta)}\}$, and the computation time gap of HJTM and ATA, i.e., $\{T_h(M, N, \eta) - T_a(M, N, \eta)\}$, for all $M$, $N$ and $\eta$ parameters, are shown in Fig. 8. ATA solves the task-to-processor allocation and the optional subtask adjustment problems in sequence, where each processor has its own frequency. Since the HJTM optimizes also the frequency-to-task assignment, and, thus, achieves a higher QoS (96.7% in average) than that of ATA. We observe that 1) the difference between $T_h(M, N, \eta)$ and $T_a(M, N, \eta)$ is small, and 2) the average value of $\{T_h(M, N, \eta)\}$ is slightly higher than $\{T_a(M, N, \eta)\}$. HJTM and ATA are both polynomial-time algorithms. Sometimes $T_h(M, N, \eta)$ can be a bit larger than $T_a(M, N, \eta)$, since 1) HJTM is a three-step heuristic while ATA is a two-step heuristic, and 2) we need to run FP method repeatedly until feasible solutions are found when solving the FAP and the TAP.

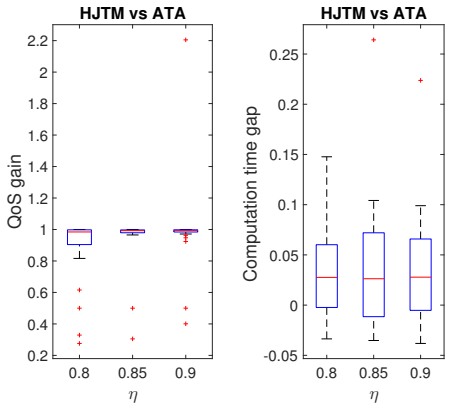



Fig. 8. QoS and computation time comparison of HJTM and ATA with $M$, $N$ and $\eta$ varying.

## VI. Conclusion

In this paper, we formulated the problem of task mapping on DVFS-enabled homogeneous multicore as an MILP, with the goal of maximizing QoS without violating the real-time and the energy supply constraints. This problem is optimally solved by the OJTM algorithm. A novel algorithm, HJTM, is proposed to further reduce the computational complexity. Our numerical results show that OJTM is guaranteed to converge to the optimal solution, while HJTM can find a feasible solution within a negligible computation time, compared with OJTM. In the future, we plan to extend our method to heterogeneous multicore platforms (e.g., big.LITTLE platforms), accelerate the OJTM and consider the dependency among the tasks.

## Acknowledgment

This research is funded by INRIA post-doctoral research fellowship program, and is partially funded by ANR ARTEFACT (AppRoximaTivE Flexible Circuits and Computing for IoT) project (Grant No. ANR-15-CE25-0015), and is partly sponsored by the National Natural Science foundation of China (Grant No. 61403340).

## References

[1] S. Girbal, M. Moretó, A. Grasset, J. Abella, and et al., "On the convergence of mainstream and mission-critical markets," in *Proc. ACM DAC*, 2013, pp. 185:1–185:10.

[2] F. Lemonnier, P. Millet, G. M. Almeida, M. Hbner, and et al., "Towards future adaptive multiprocessor systems-on-chip: an innovative approach for flexible architectures," in *Proc. IEEE SAMOS*, 2012, pp. 228–235.

[3] J. Chen, K. Hu, Q. Wang, Y. Sun, Z. Shi, and S. He, "Narrowband internet of things: Implementations and applications," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2309–2314, 2017.

[4] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Optimal reward-based scheduling for periodic real-time tasks," *IEEE Trans. Comput.*, vol. 50, no. 2, pp. 111–130, 2001.

[5] H. Chishiro and N. Yamasaki, "Practical imprecise computation model: theory and practice," in *Proc. IEEE ISORC*, 2014, pp. 198–205.

[6] J. W. S. Liu, W. K. Shih, K. J. Lin, R. Bettati, and J. Y. Chung, "Imprecise computations," *Proc. IEEE*, vol. 82, no. 1, pp. 83–94, 1994.

[7] A. Mahmood, S. A. Khan, F. Albalooshi, and N. Awwad, "Energy-aware real-time task scheduling in multiprocessor systems using a hybrid genetic algorithm," *Electron.*, vol. 6, no. 2, 2017.

[8] H. Xu, F. Kong, and Q. Deng, "Energy minimizing for parallel real-time tasks based on level-packing," in *Proc. IEEE RTCSA*, 2012, pp. 98–103.

[9] D. Li and J. Wu, "Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 810–823, 2015.

[10] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 3, pp. 111:1–111:21, 2014.

[11] F. Kong, W. Yi, and Q. Deng, "Energy-efficient scheduling of real-time tasks on cluster-based multicores," in *Proc. IEEE DATE*, 2011, pp. 1–6.

[12] L. F. Leung, C. Y. Tsui, and W. H. Ki, "Simultaneous task allocation, scheduling and voltage assignment for multiple-processors-core systems using mixed integer nonlinear programming," in *Proc. IEEE ISCAS*, 2003, pp. 309–312.

[13] L. A. Cortes, P. Eles, and Z. Peng, "Quasi-static assignment of voltages and optional cycles in imprecise computation systems with energy considerations," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 10, pp. 1117–1129, 2006.

[14] H. Yu, B. Veeravalli, and Y. Ha, "Dynamic scheduling of imprecise-computation tasks in maximizing QoS under energy constraints for embedded systems," in *Proc. IEEE ASP-DAC*, 2008, pp. 452–455.

[15] H. Yu, B. Veeravalli, Y. Ha, and S. Luo, "Dynamic scheduling of imprecise computation tasks on real-time embedded multiprocessors," in *Proc. IEEE CSE*, 2013, pp. 770–777.

[16] J. Zhou, J. Yan, T. Wei, M. Chen, and X. S. Hu, "Energy-adaptive scheduling of imprecise computation tasks for QoS optimization in real-time MPSoC systems," in *Proc. IEEE DATE*, 2017, pp. 1402–1407.

[17] T. Wei, J. Zhou, K. Cao, P. Cong, and et al., "Cost-constrained QoS optimization for approximate computation real-time tasks in heterogeneous MPSoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 1, no. 1, pp. 1–14, 2017.

[18] L. Mo, A. Kritikakou, and O. Sentieys, "Decomposed task mapping to maximize QoS in energy-constrained real-time multicores," in *Proc. IEEE ICCD*, 2017, pp. 493–500.

[19] M. Micheletto, R. Santos, and J. Orozco, "Using bioinspired metaheuristics to solve reward-based energy-aware mandatory/optional real-time tasks scheduling," in *Proc. IEEE SBESC*, 2015, pp. 132–135.

[20] M. Isabel, O. Javier, S. Rodrigo, and Z. Paula, "Energy-aware scheduling mandatory/optional tasks in multicore real-time systems," *Intl. Trans. in Op. Res.*, vol. 24, no. 12, pp. 173–198, 2017.

[21] N. W. Fisher, "The multiprocessor real-time scheduling of general task systems," *University of North Carolina at Chapel Hill*, 2007.

[22] S. Boyd, A. Ghosh, and A. Magnani, "Branch and bound methods," *Notes for EE364b, Stanford University*, pp. 1–11, 2007.

[23] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numer. Math.*, vol. 4, no. 1, pp. 238–252, 1962.

[24] W. H. Huang and J. J. Chen, "Self-suspension real-time tasks under fixed-relative-deadline fixed-priority scheduling," in *Proc. IEEE DATE*, 2016, pp. 1078–1083.

[25] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Proc. IEEE ICCAD*, 2002, pp. 721–725.

[26] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," in *Proc. ACM DAC*, 2004, pp. 275–280.

[27] H. S. Chwa, J. Seo, H. Yoo, J. Lee, and I. Shin, "Energy and feasibility optimal global scheduling framework on big.LITTLE platforms," in *Proc. IEEE RTSOPS*, 2015, pp. 1–11.

[28] H. Yu, Y. Ha, and B. Veeravalli, "Quality-driven dynamic scheduling for real-time adaptive applications on multiprocessor systems," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2026–2040, 2013.

[29] B. Samuel and N. L. Adam, "Non-convex mixed-integer nonlinear programming: a survey," *Surveys in Oper. Res. Manag. Sci.*, vol. 17, no. 2, pp. 97–106, 2012.

[30] D. McDaniel and M. Devine, "A modified Benders' partitioning algorithm for mixed integer programming," *Manag. Sci.*, vol. 24, no. 3, pp. 312–319, 1977.

[31] S. Boyd and L. Vandenberghe, "Convex optimization," *Cambridge University Press*, 2004.

[32] A. M. Geoffrion, "Generalized Benders decomposition," *J. Optim. Theory Appl.*, vol. 10, no. 4, pp. 237–260, 1972.

[33] F. Matteo, G. Fred, and L. Andrea, "The feasibility pump," *Math. Program.*, vol. 104, no. 1, pp. 91–104, 2005.

[34] J. Ramkumar and M. Tulika, "Temperature aware task sequencing and voltage scaling," in *Proc. IEEE ICCAD*, 2008, pp. 618–623.

[35] M. N. S. M. Sayuti and L. S. Indrusiak, "Real-time low-power task mapping in networks-on-chip," in *Proc. IEEE ISVLSI*, 2013, pp. 14–19.

[36] C. D. Randazzo and H. P. L. Luna, "A comparison of optimal methods for local access uncapacitated network design," *Annals of Op. Res.*, vol. 106, no. 1, pp. 263–286, 2001.

**Lei Mo** (S'13–M'17) is currently a Postdoctoral Fellow with INRIA Rennes Bretagne Atlantique research center, France. He received the B.S. degree from College of Telecom Engineering and Information Engineering, Lanzhou University of Technology, Lanzhou, China, in 2007, and the Ph.D. degree from 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 research center, France. His current research interests include networked estimation and control in wireless sensor and actuator networks, cyber-physical systems, task mapping and resources allocation in multi-core systems. He serves as an Associate Editor for KSII Transactions on Internet and Information Systems. He also served as a Guest Editor for IEEE Access and Journal of Computer Networks and Communications and TPC Member for several international conferences.

**Angeliki Kritikakou** is currently an Associate Professor at University of Rennes and IRISA–INRIA Rennes Bretagne Atlantique research center. She received her Ph.D. in 2013 from the Department of Electrical and Computer Engineering at University of Patras, Greece and in collaboration with IMEC Research Center, Belgium. She worked for one year as a Postdoctoral Research Fellow at the Department of Modelling and Information Processing (DTIM) at ONERA in collaboration with Laboratory of Analysis and Architecture of Systems (LAAS) and the University of Toulouse, France. Her research interests include embedded systems, real-time systems, mixed-critical systems, hardware/software co-design, mapping methodologies, design space exploration methodologies, memory management methodologies, low power design and fault tolerance.

**Olivier Sentieys** (M'94) is a Professor at the University of Rennes holding an INRIA Research Chair on Energy-Efficient Computing Systems. He is leading the Cairn team common to INRIA (French research institute dedicated to computational sciences) and IRISA Laboratory. He is also the head of the Computer Architecture department of IRISA. From 2012 to 2017 he was on secondment at INRIA as a Senior Research Director. His research interests are in the area of computer architectures, embedded systems and signal processing, with a focus on system-level design, energy-efficiency, reconfigurable systems, hardware acceleration, approximate computing, and power management of energy harvesting sensor networks. He authored or co-authored more than 250 journal or conference papers, holds 6 patents, and served in the technical committees of several international IEEE/ACM/IFIP conferences.