Controllable QoS for Imprecise Computation Tasks on DVFS Multicores with Time and Energy Constraints

Lei Mo, Angeliki Kritikakou, Olivier Sentieys

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

Controllable QoS for Imprecise Computation Tasks on DVFS Multicores with Time and Energy Constraints

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

***Abstract*—Multicore architectures have been used to enhance computing capabilities, but the energy consumption is still an important concern. Embedded application domains usually require less accurate, but always in-time, results. Imprecise Computation (IC) can be used to divide a task into a mandatory subtask providing a baseline QoS and an optional subtask that further increases the baseline QoS. This work aims at maximizing the system QoS by solving task mapping and DVFS for dependent IC-tasks under real-time and energy constraints. Compared with existing approaches, we consider the joint-design problem, where task-to-processor allocation, frequency-to-task assignment, task scheduling and task adjustment are optimized simultaneously. The joint-design problem is formulated as an $\mathcal{NP}$-hard Mixed-Integer Non-Linear Programming and it is safely transformed to a Mixed-Integer Linear Programming (MILP) without performance degradation. Two methods (basic and accelerated version) are proposed to find the optimal solution to MILP problem. They are based on problem decomposition and provide a controllable way to trade-off the quality of the solution and the computational complexity. The optimality of the proposed methods is proved rigorously, and the experimental results show reduced computation time (23.7% in average) compared with existing optimal methods.**

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

I. Introduction

In several application domains, less accurate results calculated in time are better solutions than accurate results calculated very late. Such domains include image and speech processing in multimedia applications, mobile target tracking, control system design etc. For instance, in control system design, the eigenvalues of the system state transfer matrix are of major importance, as they are related to the stability of system and the quality of control output. A minimum calculation accuracy is necessary. However, if increasing accuracy is possible, then the quality of the controller will be improved as well. This type of applications can be modeled as *Imprecise Computation* (IC) tasks [1], where a task is logically decomposed into a mandatory subtask and an optional subtask. The mandatory subtask must be completed before the deadline to have an acceptable result. The additional Quality of Service (QoS) is provided by the intermediate results of the optional subtask execution. The longer the execution of the optional subtasks, the higher the QoS. However, the optional subtask can be left incomplete at the cost of reduced quality.

The chip market has moved towards multi/many-core platforms [2]. It is due to the increase in system requirements, and the bottleneck of single-core systems. Such systems provide massive computing power and can thus execute a higher volume of applications in parallel. However, energy consumption has become an important concern, especially for systems with limited energy budget, such as battery-powered or energy-harvesting devices. Hence, multicore platforms have been enhanced with the capability of scaling their voltage and frequency during execution to balance system performance and energy saving. To fully exploit the features of multicore systems, mechanisms are required to decide the efficient execution of the tasks on the considered multicores, while meeting system specifications.

However, the way the IC tasks are executed on a platform is decided by several factors. The first factor is the task mapping, which refers to both the task allocation (on which core each task is executed) and the task scheduling (when each task starts execution). The second factor is the decision of the voltage and frequency of the core when it runs a specific task. The third factor is the adjustment of the optional part of each task. State-of-the-art task mapping approaches focus on how to minimize energy consumption under real-time constraints (energy-aware task mapping). Usually they are based on Dynamic Voltage and Frequency Scaling (DVFS) techniques, whereas the task model does not consider imprecise computation. Other approaches focus on how to maximize the QoS under energy and real-time constraints (QoS-aware task mapping). However, they focus on single core and multicore platforms without DVFS capabilities, whereas the task model is simplified considering independent tasks. The QoS-aware task mapping for multicore platforms is still an open issue, since there is no optimal polynomial-time solution [3].

In this work, we combine task mapping, DVFS and task adjustment for tasks with dependences under the IC model in order to: 1) maximize QoS by determining how many optional subtasks should be executed, and 2) satisfy the energy and real-time constraints by allocating and scheduling the tasks onto the cores and deciding the voltage/frequency among different tasks.

A. Related Work

1) Energy-aware Task Mapping: Several previous research contributions have addressed the energy-aware task mapping based on DVFS. When considering the independent real-time tasks and the multicore platforms, task allocation with DVFS is formulated as an Integer Linear Programming (ILP) problem in [4], as the frequency assigned to each processor is fixed. Therefore, a Linear Programming (LP)-based relaxation algorithm is proposed to solve this problem. Combining DVFS and Dynamic Power Management (DPM), a Mixed-Integer Linear Programming (MILP)-based task mapping problem is formulated in [5]. The number of variables is further reduced by problem refinement, and then the refined MILP problem is optimally solved by the CPLEX solver.

Considering dependent real-time tasks adds another dimension to be optimized in the problem under study (i.e., the

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 has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

sequence between the tasks). In [6]–[8], ILP is used to formulate the voltage selection/task allocation problem. To solve the ILP problem, an LP-based heuristic method is proposed in [6], a hybrid Genetic Algorithm (GA) is designed in [7], and an optimal Benders decomposition-based method is presented in [8]. The work in [9] considers an MILP-based task mapping problem and this problem is optimally solved by the CPLEX solver. The Mixed-Integer Non-Linear Programming (MINLP)-based task mapping problem in [10] is first relaxed to an MILP by linear approximation and then is solved by the Branch and Bound (B&B) method.

Besides minimizing computation energy, the communication energy is also taken into account in [11]. The related problem is formulated by a Convex Non-Linear Programming (NLP) and an ILP-based algorithm is proposed as a solution. Some works also consider bi-objective optimization, e.g., minimizing the energy consumption/makespan (overall execution time of the tasks) and maximizing the reliability of system [12], [13]. As long as the constraints (e.g., deadline) of the tasks allows it, the aforementioned methods achieve significant energy reductions compared to the reference scenario where no DVFS is applied. However, they do not focus on QoS related issues.

2) QoS-aware Task Mapping: Other approaches aim at maximizing QoS for IC-tasks under energy supply and real-time constraints. Note that the aim of DPM is to explore the idle intervals of a processor and switch the processor to a sleep mode when the idle interval exceeds a certain level [5]. The approaches that focus on QoS-aware task mapping are extensions of DVFS. Several works have been presented for single processors. Hence, they focus on task scheduling and optional subtasks adjustment, as there is no need to consider task allocation. For instance, the work in [14] considers the dependent task model. It formulates the task scheduling problem with DVFS as an NLP, and a quasi-static approach is proposed as a solving method. A similar problem is considered in [15], [16], but tasks are considered independent.

For multicore platforms, most of the related work is on independent task model. In [17], [18], the task mapping of independent tasks is considered, but each processor has decided its own frequency upfront, whereas task allocation and optional subtasks adjustment sub-problems are solved in sequence. Previous work in [19] considers independent tasks and no DVFS capabilities. The work in [20] focuses on the task scheduling and optional subtasks adjustment of dependent tasks, since the task-processor binding is given upfront. Except 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 [3], [21]. All the aforementioned QoS-aware task mapping methods, except [14], [19], employ heuristics to find the near-optimal solutions. Although heuristics can provide feasible solutions in a short amount of time, they do not provide any bounds on solution quality, and are sensitive to changes in the problem structure and parameters.

B. Contributions

Our goal is to solve the joint-design problem of QoS-aware task mapping and DVFS for real-time dependent IC-tasks on multicore platforms. We determine on which processor should the task be executed (task-to-processor allocation), its frequency (frequency-to-task assignment), the start time (task scheduling), and the end time (task adjustment) of each task, such that the system QoS is maximized while meeting the energy supply and the deadline of each task. The two main differences of this work with the state-of-the-art are that: i) task-to-processor allocation, frequency-to-task assignment, task scheduling and task adjustment are optimized simultaneously (joint-design problem). Solving these correlated sub-problems separately may lead to non-optimal solutions; ii) although the joint-design problem is complex, we propose a method to solve it using polynomial-time algorithms, and we can thus obtain an optimal solution while avoiding high computational complexity. To achieve that, the joint-design problem is initially formulated as an $\mathcal{NP}$-hard MINLP and then it is safely transformed to an MILP without performance degradation. Further, the MILP problem is optimally solved by iterating Linear Programming (LP)-based subproblems.

Compared with previous work in [19], this paper considers dependent tasks (each task has its own deadline) and DVFS-enabled multicore platforms. First, the problem formulation is different, as the joint-design problem (task-to-processor allocation, frequency-to-task assignment, task scheduling and task adjustment) is more complex than the previously published work. The previous work in [19] only considers task-to-processor allocation and task adjustment, and the related problem is formulated as an MILP. In contrast, the joint-design problem considered in this paper is an MINLP. Second, based on this new formulation, two algorithms are proposed in this paper which are able to find an optimal solution. Besides the basic method which is proposed in the previously published work, this paper also presents a way to accelerate the basic method.

Our main contributions are summarized as follows:

1) The joint-design problem to maximize system QoS, while the real-time and energy consumption constraints are not violated, is formulated as an MINLP problem.

2) We prove that by introducing the auxiliary variables and the additional constraints, the nonlinear items, which are introduced by the coupling of frequency-to-task assignment and task adjustment, can be linearized. As this linearization does not change the objective function and the feasible region of the problem, the MINLP problem is safely transformed to an MILP problem.

3) A novel Joint DVFS-QoS-aware Task mapping (JDQT) algorithm is proposed to optimally solve the MILP problem. The basic idea of JDQT algorithm is similar to closed-loop control and it is based on Benders decomposition. The MILP problem is decomposed into two smaller sub-problems with fewer variables and constraints: an ILP-based Master Problem (MP) for task-to-processor allocation and frequency-to-task assignment decisions, and an LP-based Slave Problem (SP) for task scheduling and task adjustment decisions. We prove that: i) at each iteration a lower bound and an upper bound are obtained by solving the SP and the MP, respectively, and ii) adding a proper constraint, which is constructed by the solution of the SP at previous iteration, into the MP at current iteration and solving the updated MP and the SP iteratively, the gap between the lower and the upper bounds converges to a predefined threshold within a finite number of iterations.

4) A novel Accelerated JDQT (AJDQT) algorithm is proposed to further reduce the computing time of JDQT algorithm, without violating optimality of the solution. The AJDQT algorithm is based on the fact that the computational complexity of JDQT algorithm is dominated by the cost of solving the ILP-based MP. Hence, we relax the MP to an LP to find a feasible solution. We prove that by replacing the optimal solution of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

MP with the feasible solution during the iteration between the MP and the SP, the optimality of the solution is guaranteed, since the additional constraints added into the MP only exclude the non-optimal and infeasible solutions. Moreover, based on the structure of the relaxed MP and SP, we design a two-layer subgradient-based algorithm to solve these problems in a distributed manner. We prove that this method is able to find an optimal solution and can further reduce the computing time of the obtained solution due to its distributed structure.

5) As the proposed JDQT and AJDQT algorithms run in an iterative way, the stopping criteria of JDQT and AJDQT algorithms can serve as the controllable parameters to trade-off the quality of the solution (i.e., system QoS) and the computational complexity (i.e., computing time).

6) Finally, we provide extensive experimental results to analyze the quality of the solution, the computing time and the scalability of the proposed JDQT and AJDQT algorithms and compare it with stochastic methods, optimal methods and heuristics. The obtained results show that the proposed methods achieve optimal solution with reduced computing time compared to state-of-the-art optimal methods. In addition, we present how we can further reduce the computing time by controlling the solution quality.

C. Paper Organization

The paper is organized as follows: Section II presents the system model and problem formulation. In Section III, we present and formulate the JDQT and AJDQT algorithms. Section IV evaluates the performance of the proposed algorithms through several experimental results. Finally, we conclude this study in Section V.

II. System Model and Problem Formulation

A. System Model

1) Task Model: We consider a set of dependent real-time tasks $\mathcal{T} \triangleq \{\tau_1, \ldots, \tau_N\}$. The task set is modeled by a Directed Acyclic Graph (DAG) called $G(V, E)$, where vertexes V denote the set of tasks to be executed, while edges E describe the data dependencies between the tasks. Without loss of generality, it is assumed that all tasks are released at the same time 0. A task starts its execution when all its preceding tasks have been completed. Each task is executed by only one processor with no preemption, such that multiple tasks can be executed on the same processor in non-overlapped time slots.

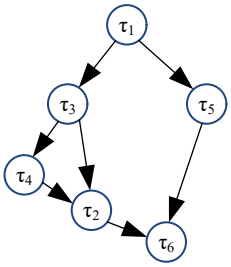

Fig. 1. Task graph of illustration example.

Task τ_i is described by a tuple $\{o_i, M_i, O_i, t_i^s, d_i, T_i\}$, where M_i is the mandatory subtask, o_i is the optional subtask that can be executed to improve the result of mandatory subtask. The optional subtask has an upper bound O_i (i.e., $0 \leq o_i \leq O_i$). M_i and O_i are measured in Worst Case Execution Cycles (WCECs) [16]. t_i^s and d_i represent the start time and the deadline of task τ_i, respectively. Note that t_i^s is an unknown variable, which is determined by task mapping decision. T_i is the period of task τ_i. H is the hyper-period of the task set $\mathcal{T}$ (i.e. the least common multiple of $\{T_1, \ldots, T_N\}$). The sequences between the tasks in one hyper-period H can be formulated by an $N \times N$ binary matrix $\boldsymbol{S}$. Denote s_{ij} as the $(i,j)^{th}$ element of matrix $\boldsymbol{S}$. If task τ_i precedes task τ_j (i.e., τ_j starts after the end time of τ_i), $s_{ij} = 1$, otherwise, $s_{ij} = 0$ $(i \neq j)$. Fig. 1 shows the DAG of an illustration example. The task sequence matrix associated with this DAG is given by

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

2) Platform Model: We consider a multicore platform with M homogeneous DVFS-enabled processors $\{\theta_1, \ldots, \theta_M\}$. Each processor θ_k has L different Voltage/Frequency (V/F) levels $\{(v_1, f_1), \ldots, (v_L, f_L)\}$. The power of the processor under the given V/F level (v_l, f_l) is computed as

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

where P_l^s is the static power, and P_l^d is the dynamic power. The dynamic power P_l^d is related to the clock frequency f_l and supply voltage v_l of the processor, expressed as $P_l^d = C_{eff} v_l^2 f_l$, where C_{eff} is the average effective switching capacitance. For the static power P_l^s, we adopt $P_l^s = v_l K_1 e^{K_2 v_l} e^{K_3 v_{bs}} + |v_{bs}| I_j$ from [22]. The constants K_1, K_2 and K_3 are technology dependent. 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 [23]. The energy model described by (1) is widely adopted by similarly published works [5], [17], [20].

We focus on inter-task DVFS [5], [16] where the frequency of the processor stays constant during the execution of a task. It is assumed that when a processor does not execute a task, it goes to the idle state. This transition time and energy overhead is very small compared to the time and energy required to complete a task, and it is assumed to be incorporated into the execution time and energy of the task [4]. The system is energy-constrained in the sense that it has a fixed energy budget E_s which cannot be replenished during the hyper-period H [17]. Taking the available energy E_s into account, the system operation can be divided into three states: i) *Low*: the supplied energy E_s is insufficient to execute all the mandatory subtasks $\{M_1, \ldots, M_N\}$, ii) *High*: the supplied energy E_s is sufficient to execute all the mandatory and optional subtasks $\{M_1 + O_1, \ldots, M_N + O_N\}$, and iii) *Medium*: all the mandatory subtasks are ensured to finish, while not all the optional subtasks have enough energy to complete their executions. This work focuses on the medium state.

B. Problem Formulation

The problem consists of an objective function to maximize the QoS of the system subject to a set of real-time and energy constraints. In this context, we determine i) on which processor should the task be executed (*task-to-processor allocation*), ii) what V/F should be used for the task (*frequency-to-task assignment*), iii) when should the task starts (*task scheduling*), and iv) how many and how long are the optional subtasks (*task adjustment*). Therefore, we define the following variables:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

i) binary variable $q_{ik} = 1$ if task τ_i runs on processor θ_k, otherwise, $q_{ik} = 0$, ii) binary variable $c_{il} = 1$ if task τ_i executes with frequency f_l, otherwise, $c_{il} = 0$, iii) continuous variable t_i^s denotes the start time of task τ_i, and iv) continuous variable o_i represents the optional subtask of task τ_i.

We consider o_i as continuous variables, and then we round the result down. This impact of one cycle is negligible since the tasks execute typically hundreds of thousands of cycles [14]. In order to formulate the joint-design problem, we have to cope with the constraints of i) task non-preemption, ii) task dependency, iii) task deadline, and iv) energy supply, simultaneously. Let $\mathcal{N} \triangleq \{1, \ldots, N\}$, $\mathcal{M} \triangleq \{1, \ldots, M\}$ and $\mathcal{L} \triangleq \{1, \ldots, L\}$. We present our problem formulation as follows.

1) Task-to-Processor Allocation and Frequency-to-Task Assignment Constraints: Since i) each task can be assigned only one V/F, and ii) each task is executed on only one processor, the task allocation and frequency assignment variables c_{il} and q_{ik} must satisfy:

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

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

2) Task Non-Preemption Constraints: Since each processor executes no more than one task at the same time, the non-preemptive constraint requires that no task executed in the same processor can overlap with each other. Hence, we have

$$t_i^e \le t_j^s + (2 - q_{ik} - q_{jk})H + (1 - s_{ij})H, \quad \forall i, j \in \mathcal{N}, \ i \ne j, \ \forall k \in \mathcal{M}, \tag{4}$$

$$t_j^e \le t_i^s + (2 - q_{ik} - q_{jk})H + s_{ij}H, \quad \forall i, j \in \mathcal{N}, \ i \ne j, \ \forall k \in \mathcal{M}, \tag{5}$$

where $t_i^e = t_i^s + \sum_{l=1}^{L} c_{il}(w_{il} + \frac{o_i}{f_l})$ is the end time of task τ_i, and $w_{il} = \frac{M_i}{f_l}$ is the execution time of mandatory subtask M_i with frequency f_l.

Constraint (4) guarantees that task τ_j starts running only after task τ_i is finished. Whereas constraint (5) describes the other case (i.e., task τ_j should be finished before the start of task τ_i). If tasks τ_i and τ_j are executed on the same processor (i.e., $q_{ik} = q_{jk} = 1$), constraints (4) and (5) are meaningful, else, constraints (4) and (5) are always satisfied. Assume that $q_{ik} = q_{jk} = 1$ right now. If task τ_i precedes task τ_j (i.e., $s_{ij} = 1$), we have constraint $t_i^e \le t_j^s$, and, thus, the constraint $t_j^e \le t_i^s + H$ is always satisfied. On the other hand, if $s_{ij} = 0$, the constraint $t_j^e \le t_i^s$ is sufficient, as the constraint $t_i^e \le t_j^s + H$ is always satisfied. Note that if task set $\mathcal{T}$ exists independent tasks, constraint (5) can be removed.

3) Task Dependency Constraints: One challenge to determine the number of cycles assigned to optional subtasks $\{o_1, \ldots, o_N\}$ is that the interval of two adjacent tasks is unknown, since the start time of the tasks is the variable. To illustrate this challenge, we use our illustration example in Fig. 1, where the tasks are allocated to four cores. Fig. 2 shows the corresponding task execution sequence. We can see that τ_5 is the closest task after task τ_1. Hence, the start time of tasks τ_1 and τ_5 and the execution time of task τ_1 are highly correlated with each other. To determine the execution sequences between the tasks, based on the task sequence matrix $\boldsymbol{S}$, we introduce an Execution Order Decision (EOD) matrix $\boldsymbol{P}$ [5]. Denote p_{ij} as the $(i,j)^{th}$ element of matrix $\boldsymbol{P}$. If task τ_i precedes task τ_j and task τ_j is the closest task to task τ_i, $p_{ij} = 1$, otherwise, $p_{ij} = 0$.

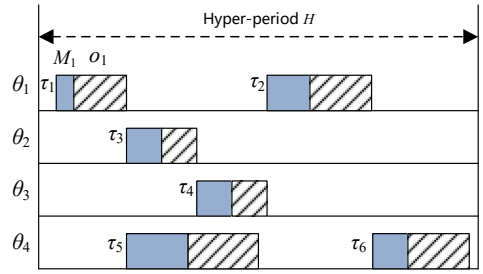

Fig. 2. Task execution sequence example corresponding to task sequence $\boldsymbol{S}$ defined in Fig. 1.

The EOD matrix associated with the illustration example in Fig. 1 is given by

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

According to the non-zero elements in matrix $\boldsymbol{P}$, we obtain a task execution sequence $\{\tau_1 \rightarrow \{\overset{\tau_3 \rightarrow \tau_4 \rightarrow \tau_2}{\tau_5}\} \rightarrow \tau_6\}$. Therefore, tasks τ_5 and $\{\tau_3, \tau_4, \tau_2\}$ can be executed in different cores simultaneously since they are independent.

Based on the EOD matrix $\boldsymbol{P}$, the constraints on the start time and the end time of tasks τ_i and τ_j are

$$t_j^s + (1 - p_{ij})H \ge t_i^s + p_{ij} \sum\nolimits_{l=1}^{L} c_{il}\Big(w_{il} + \frac{o_i}{f_l}\Big), \quad \forall i, j \in \mathcal{N}, \ i \ne j, \tag{6}$$

$$t_j^s - (1 - p_{ij})H \le t_i^s + p_{ij} \sum\nolimits_{l=1}^{L} c_{il}\Big(w_{il} + \frac{o_i}{f_l}\Big), \quad \forall i, j \in \mathcal{N}, \ i \ne j. \tag{7}$$

If $p_{ij} = 1$, task τ_i precedes task τ_j and τ_j is the closest task of task τ_i. Constraints (6) and (7) will be meaningful for task start time t_i^s and t_j^s (i.e., $t_j^s = t_i^s + \sum_{l=1}^{L} c_{il}(w_{il} + \frac{o_i}{f_l})$). While for $p_{ij} = 0$, constraints (6) and (7) can be ignored since the inequalities $t_j^s + H \ge t_i^s$ and $t_j^s - H \le t_i^s$ are always true.

4) Task Deadline Constraints: Since task τ_i should be executed within a time threshold d_i $(0 \le d_i \le H)$, the task real-time constraint is given by

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

5) Energy Supply Constraint: Based on the energy model described by (1), the total energy consumed by M processors during the hyper-period H is bounded by

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

Since the system QoS derives from executing the optional subtasks $\{o_1, \ldots, o_N\}$, we introduce a QoS function [17], [18], defined as $\sum_{i=1}^{N} o_i$, and our aim is to maximize it (or to minimize its negative). Based on this objective function and the aforementioned constraints, the Primal Problem (PP) is formulated as

$$\mathbf{PP}: \min_{\boldsymbol{c},\boldsymbol{q},\boldsymbol{o},\boldsymbol{t}^s} \Phi = -\sum_{i=1}^{N} o_i \quad (10)$$
$$\text{s.t.} \begin{cases} (2)-(9),\ c_{il}, q_{ik} \in \{0,1\},\ 0 \le t_i^s \le H, \\ 0 \le o_i \le O_i,\ \forall i \in \mathcal{N},\ \forall l \in \mathcal{L},\ \forall k \in \mathcal{M}. \end{cases}$$

Due to the product $c_{il}o_i$ of the optimization variables in the constraints (4) – (9), the PP (10) is an MINLP problem.

Theorem 2.1: The QoS-aware IC-task mapping problem based on DVFS (i.e., PP (10)) is $\mathcal{NP}$-hard.

Proof: Please refer to [24] for the details. ■

III. MILP-based Approach

This section presents the proposed MILP formulation to linearize the MINLP problem and the two proposed algorithms, i.e. the optimal JDQT and the Accelerated JDQT algorithms.

A. MINLP linearization

Note that as the PP (10) is an MINLP problem, solving it is much more time consuming than solving an MILP problem [10]. In this section, we adopt the idea of *variable replacement* [5], [25] to eliminate the nonlinear item $c_{il}o_i$. By doing so, the MINLP can be safely transformed to an MILP without performance degradation.

Lemma 3.1: Given constants $s_1, s_2 > 0$ 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 $-s_1 \le x \le s_2$. For the same reason, if $b = 1$, we have $-s_1 \le h = x \le s_2$. $S_2 \Rightarrow S_1$ holds. ■

Since variables $c_{il} \in \{0,1\}$ and $0 \le o_i \le O_i$, based on *Lemma 3.1*, an *auxiliary variable* (continuous) h_{il} is introduced to replace the nonlinear item $c_{il}o_i$. Therefore, the constraints (4) – (9) are safely replaced by the following constraints:

$$t_i^s + \sum_{l=1}^{L} \left(c_{il} w_{il} + \frac{h_{il}}{f_l} \right) \le t_j^s + (2 - q_{ik} - q_{jk}) H + (1 - s_{ij}) H,\ \forall i,j \in \mathcal{N},\ i \ne j,\ \forall k \in \mathcal{M}, \quad (11)$$

$$t_j^s + \sum_{l=1}^{L} \left(c_{jl} w_{jl} + \frac{h_{jl}}{f_l} \right) \le t_i^s + (2 - q_{ik} - q_{jk}) H + s_{ij} H,\ \forall i,j \in \mathcal{N},\ i \ne j,\ \forall k \in \mathcal{M}, \quad (12)$$

$$t_j^s + (1 - p_{ij}) H \ge t_i^s + p_{ij} \sum_{l=1}^{L} \left(c_{il} w_{il} + \frac{h_{il}}{f_l} \right),\ \forall i,j \in \mathcal{N},\ i \ne j, \quad (13)$$

$$t_j^s - (1 - p_{ij}) H \le t_i^s + p_{ij} \sum_{l=1}^{L} \left(c_{il} w_{il} + \frac{h_{il}}{f_l} \right),\ \forall i,j \in \mathcal{N},\ i \ne j, \quad (14)$$

$$t_i^s + \sum_{l=1}^{L} \left(c_{il} w_{il} + \frac{h_{il}}{f_l} \right) \le d_i,\ \forall i \in \mathcal{N}, \quad (15)$$

$$\sum_{i=1}^{N} \sum_{l=1}^{L} \left(c_{il} w_{il} + \frac{h_{il}}{f_l} \right) (P_l^d + P_l^s - P_0^s) + MHP_0^s \le E_s, \quad (16)$$

$$h_{il} \le c_{il} O_i,\ \forall i \in \mathcal{N},\ \forall l \in \mathcal{L}, \quad (17)$$

$$h_{il} \le o_i,\ \forall i \in \mathcal{N},\ \forall l \in \mathcal{L}, \quad (18)$$

$$h_{il} + (1 - c_{il}) O_i \ge o_i,\ \forall i \in \mathcal{N},\ \forall l \in \mathcal{L}. \quad (19)$$

where (17) – (19) are the additional constraints introduced by the linearization. Consequently, the PP (10) is rewritten as

$$\mathbf{PP1}: \min_{\substack{\boldsymbol{c},\boldsymbol{q},\boldsymbol{o},\\ \boldsymbol{t}^s,\boldsymbol{h}}} \Phi = -\sum_{i=1}^{N} o_i \quad (20)$$
$$\text{s.t.} \begin{cases} (2),(3),(11)-(19),\ c_{il}, q_{ik} \in \{0,1\},\ 0 \le t_i^s \le H, \\ 0 \le o_i, h_{il} \le O_i,\ \forall i \in \mathcal{N},\ \forall l \in \mathcal{L},\ \forall k \in \mathcal{M}. \end{cases}$$

The PP1 (20) is an MILP problem, as the binary and continuous variables are coupled with each other linearly. Hence, it is much easier to solve than the PP (10). Furthermore, *Lemma 3.1* implies that the variable replacement will 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 PP and PP1 are the same).

B. Basic Solution Method: JDQT

In this section, we propose a novel Joint DVFS QoS-aware Task mapping (JDQT) algorithm, to efficiently solve the PP1 (20). The structure of JDQT is shown in Fig. 3. To solve the PP1, finding the optimal task-to-processor allocation and frequency-to-task assignment decisions is the most important step. This is because if binary variables $\boldsymbol{c}$ and $\boldsymbol{q}$ are determined, the PP1 will reduce to an LP problem, which has a simpler structure and is much easier to solve. Benders decomposition [26], [27] is an effective method for solving MILP with guaranteed global optimality. Instead of considering all the variables and the constraints of PP1 simultaneously, Benders decomposition divides the PP1 into two smaller subproblems with fewer variables and constraints (i.e. *Master Problem (MP)* and *Slave Problem (SP))*. Then, it solves the subproblems iteratively by utilizing the solution of one in the other.

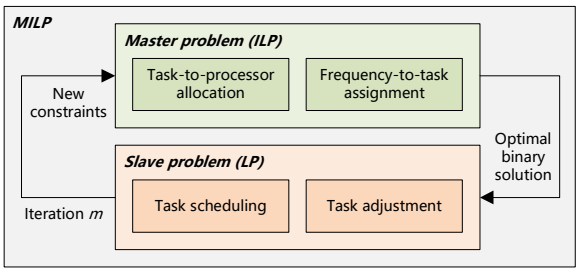

Fig. 3. The structure of JDQT algorithm.

The MP involves all the binary variables $\{\boldsymbol{c}, \boldsymbol{q}\}$ and the corresponding part of the objective function, as well as the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

constraints of PP1. It also contains one auxiliary (continuous) variable $\hat{\Phi}$ that is introduced to facilitate the interaction between the MP and the SP. On the other hand, the SP incorporates all the continuous variables $\{\boldsymbol{o}, \boldsymbol{t}^s, \boldsymbol{h}\}$ and the associated constraints of PP1. Initially, we solve the MP and obtain a lower bound for PP1's optimal objective function value along with a set of values for the binary variables. We then substitute these values into the SP and solve the dual of the SP (DSP) to obtain an upper bound for PP1's optimal objective function value. Based on the solution of the DSP, a new constraint called *Benders cut* is generated. This constraint is added into the MP, and a new iteration is performed to solve the updated MP and SP. The iteration process stops when the gap between the upper and lower bounds is smaller than a predefined threshold.

1) The MP and The SP: Based on the structure of PP1 (20), the corresponding MP and SP are formulated as follows:

$$\textbf{MP}: \Phi_l(m) = \min_{\hat{\Phi}, \boldsymbol{x}} \hat{\Phi} \tag{21}$$
$$\text{s.t.} \begin{cases} (2), (3), \\ \hat{\Phi} \geq \mathcal{F}(\boldsymbol{x}, \boldsymbol{\omega}(\varsigma)),\ \forall \varsigma \in \mathcal{A}, \\ 0 \geq \mathcal{F}(\boldsymbol{x}, \boldsymbol{\nu}(\vartheta)),\ \forall \vartheta \in \mathcal{B}, \\ c_{il}, q_{ik} \in \{0,1\}, \hat{\Phi} \geq 0, \forall i \in \mathcal{N}, \forall l \in \mathcal{L}, \end{cases}$$

where function $\mathcal{F}(\boldsymbol{x}, \boldsymbol{\omega})$ is given by

$$\begin{aligned} &\mathcal{F}(\boldsymbol{x}, \boldsymbol{\omega}) = \\ &\sum\nolimits_{i=1, i\neq j}^{N} \sum\nolimits_{j=1}^{N} \Big\{ \Big[\sum\nolimits_{k=1}^{M} (\alpha_{ij}^k + \beta_{ij}^k) + (\gamma_{ij} - \delta_{ij}) p_{ij} \Big] \cdot \\ &\sum\nolimits_{l=1}^{L} c_{il} w_{il} - \sum\nolimits_{k=1}^{M} \{ (\alpha_{ij}^k + \beta_{ij}^k)(2 - q_{ik} - q_{jk}) \\ &+ [\alpha_{ij}^k (1 - s_{ij}) + \beta_{ij}^k s_{ij}] \} H - (\gamma_{ij} + \delta_{ij})(1 - p_{ij}) H \Big\} \\ &+ \sum\nolimits_{i=1}^{N} \sum\nolimits_{l=1}^{L} \{ [c_{il} w_{il} (P_l^d + P_l^s - P_0^s) \\ &+ MHP_0^s - E_s] \varphi - [(\theta_{il} + \tau_{il}) c_{il} + \tau_{il}] O_i \}. \end{aligned}$$

In MP (21), $\boldsymbol{x} \triangleq (\boldsymbol{c}, \boldsymbol{q})$ are the binary variables, $\boldsymbol{\omega}(m) \triangleq (\boldsymbol{\alpha}(m), \boldsymbol{\beta}(m), \boldsymbol{\gamma}(m), \boldsymbol{\delta}(m), \boldsymbol{\varepsilon}(m), \varphi(m), \boldsymbol{\theta}(m), \boldsymbol{\lambda}(m), \boldsymbol{\tau}(m))$ is the solution of the DSP (27) at the m^{th} iteration, and $\boldsymbol{\nu}(m) \triangleq (\hat{\boldsymbol{\alpha}}(m), \hat{\boldsymbol{\beta}}(m), \hat{\boldsymbol{\gamma}}(m), \hat{\boldsymbol{\delta}}(m), \hat{\boldsymbol{\varepsilon}}(m), \hat{\varphi}(m), \hat{\boldsymbol{\theta}}(m), \hat{\boldsymbol{\lambda}}(m), \hat{\boldsymbol{\tau}}(m))$ is the solution of the Dual Feasibility Check Problem (DFCP) (33) at the m^{th} iteration. Replacing $\boldsymbol{\omega}$ in $\mathcal{F}(\boldsymbol{x}, \boldsymbol{\omega})$ with $\boldsymbol{\nu}$, we obtain $\mathcal{F}(\boldsymbol{x}, \boldsymbol{\nu})$. Items $\hat{\Phi} \geq \mathcal{F}(\boldsymbol{x}, \boldsymbol{\omega}(\varsigma))$ and $0 \geq \mathcal{F}(\boldsymbol{x}, \boldsymbol{\nu}(\vartheta))$ are the feasibility and infeasibility constraints (Benders cuts), respectively. They are generated from the solution of the DSP. $\mathcal{A}$ and $\mathcal{B}$ are the sets of iterations that the DSP has the bounded and unbounded solutions, respectively. Since i) the objective function of PP1 is constructed by the continuous variables $\boldsymbol{o}$, and ii) the MP only considers the binary variables $\boldsymbol{x}$, we introduce an *auxiliary variable* (continuous) $\hat{\Phi}$ for the MP as the objective function. $\hat{\Phi}$ is a proxy of PP1's objective function Φ, and both have the same physical meaning. We explain the relationship between them in the next section.

$$\textbf{SP}: \Phi_u(m) = \min_{\boldsymbol{y} \succeq 0}\ \Phi \tag{22}$$
$$\text{s.t.}\ \ (11) - (19)\ with\ \boldsymbol{x}(m),$$

where $\boldsymbol{y} \triangleq (\boldsymbol{o}, \boldsymbol{t}^s, \boldsymbol{h})$ are the continuous variables, and $\boldsymbol{x}(m)$ is the solution of the MP at the m^{th} iteration. In fact, the SP (22) is "identical" to the PP1 (20). Their formulations are the same except that the binary variables $\boldsymbol{x}(m)$ in the SP are fixed and given in advance.

2) The Iterations Between The MP and The SP: Note that the MP (21) only contains the task-to-processor allocation and the frequency-to-task assignment decisions (i.e., $\boldsymbol{x}$). Compared with the PP1 (20), the task scheduling and adjustment related constraints are relaxed, thus solving the MP yields a *lower bound* $\Phi_l(m)$ (the proof is provided in *Lemma 3.2*). On the other hand, since i) $\boldsymbol{x}(m)$ may be just a feasible solution, and ii) the PP1 is a minimization problem, solving the SP with the non-optimal decision $\boldsymbol{x}(m)$ yields an *upper bound* $\Phi_u(m)$. Denote $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ as the optimal solution of PP1, and Φ^* as the corresponding optimal objective function value. Hence, we have $\Phi_u(m) \leq \Phi^* \leq \Phi_l(m)$.

Definition 3.1: (*ϵ-optimal solution*) Denote $(\boldsymbol{x}(m), \boldsymbol{y}(m))$ as an arbitrary solution of the PP1 (20), and $\Phi(m)$ as the corresponding objective function value. If $|\Phi(m) - \Phi^*| \leq \epsilon$, we say $(\boldsymbol{x}(m), \boldsymbol{y}(m))$ is an ϵ-optimal solution.

At each iteration, there is always a new feasibility constraint (31) or infeasibility constraint (34) added into the MP (21) to raise the lower bound $\Phi_l(m)$ and reduce the upper bound $\Phi_u(m)$. If $|\Phi_u(m) - \Phi_l(m)| \leq \epsilon$, $(\boldsymbol{x}(m), \boldsymbol{y}(m))$ is an ϵ-optimal solution. And further, if ϵ is a small positive value, the optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is found. The iteration process is summarized as follows:

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. The initial solution $\boldsymbol{x}(0)$ can be given arbitrarily, as long as it satisfies the constraints (2) – (3).

Step 2 – Solving Slave Problem: In this paper, rather than solving the SP (22) directly, we solve its dual problem. This is because i) the SP is a LP problem and the optimal objective function values of the SP and its dual problem are equivalent due to the strong duality [28], ii) we can construct the feasibility and infeasibility constraints through the solution of the DSP, and iii) the dual problem has fewer number of constraints than its primal problem.

To construct the dual problem of SP, we introduce the positive Lagrange multipliers $\boldsymbol{\alpha} \triangleq [\boldsymbol{\alpha}^k]$ ($\boldsymbol{\alpha}^k \triangleq [\alpha_{ij}^k]$), $\boldsymbol{\beta} \triangleq [\boldsymbol{\beta}^k]$ ($\boldsymbol{\beta}^k \triangleq [\beta_{ij}^k]$), $\boldsymbol{\gamma} \triangleq [\gamma_{ij}]$, $\boldsymbol{\delta} \triangleq [\delta_{ij}]$, $\boldsymbol{\varepsilon} \triangleq [\varepsilon_i]$, φ, $\boldsymbol{\theta} \triangleq [\theta_{il}]$, $\boldsymbol{\lambda} \triangleq [\lambda_{il}]$, $\boldsymbol{\tau} \triangleq [\tau_{il}]$ ($\forall i, j \in \mathcal{N}$, $\forall l \in \mathcal{L}$, $\forall k \in \mathcal{M}$) to the constraints (11) – (19). The corresponding Lagrangian is

$$\begin{aligned} &\mathcal{L}(\boldsymbol{x}(m), \boldsymbol{y}, \boldsymbol{\omega}) \\ &= -\sum\nolimits_{i=1}^{N} o_i + \sum\nolimits_{i=1, i\neq j}^{N} \sum\nolimits_{j=1}^{N} \sum\nolimits_{k=1}^{M} \Big\{ \alpha_{ij}^k \Big[t_i^s - t_j^s \\ &+ \sum\nolimits_{l=1}^{L} \Big(c_{il}(m) w_{il} + \frac{h_{il}}{f_l} \Big) - (2 - q_{ik}(m) - q_{jk}(m)) H \\ &- (1 - s_{ij}) H \Big] + \beta_{ij}^k \Big[t_j^s + \sum\nolimits_{l=1}^{L} \Big(c_{il}(m) w_{il} + \frac{h_{il}}{f_l} \Big) - t_i^s \\ &- (2 - q_{ik}(m) - q_{jk}(m)) H - s_{ij} H \Big] \Big\} \\ &+ \sum\nolimits_{i=1, i\neq j}^{N} \sum\nolimits_{j=1}^{N} \Big\{ \gamma_{ij} \Big[p_{ij} \sum\nolimits_{l=1}^{L} \Big(c_{il}(m) w_{il} + \frac{h_{il}}{f_l} \Big) \\ &+ t_i^s - t_j^s - (1 - p_{ij}) H \Big] + \delta_{ij} \Big[t_j^s - (1 - p_{ij}) H - t_i^s \\ &- p_{ij} \sum\nolimits_{l=1}^{L} \Big(c_{il}(m) w_{il} + \frac{h_{il}}{f_l} \Big) \Big] \Big\} \\ &+ \sum\nolimits_{i=1}^{N} \varepsilon_i \Big[t_i^s + \sum\nolimits_{l=1}^{L} \Big(c_{il}(m) w_{il} + \frac{h_{il}}{f_l} \Big) - d_i \Big] \end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

$$+\varphi\Big[\sum\nolimits_{i=1}^{N}\sum\nolimits_{l=1}^{L}\Big(c_{il}(m)w_{il}+\frac{h_{il}}{f_l}\Big)(P_l^d+P_l^s-P_0^s)$$
$$+MHP_0^s-E_s\Big]+\sum\nolimits_{i=1}^{N}\sum\nolimits_{l=1}^{L}[\theta_{il}(h_{il}-c_{il}(m)O_i)$$
$$+\lambda_{il}(h_{il}-o_i)+\tau_{il}(o_i-h_{il}-(1-c_{il}(m))O_i)].$$

The dual function $\mathcal{R}(\boldsymbol{x}(m),\boldsymbol{\omega})$ is defined as the minimum value of the Lagrangian $\mathcal{L}(\boldsymbol{x}(m),\boldsymbol{y},\boldsymbol{\omega})$ with respective to the variables $\boldsymbol{y}$ [28], i.e., under the given $\boldsymbol{\omega}$,

$$\mathcal{R}(\boldsymbol{x}(m),\boldsymbol{\omega})=\min_{\boldsymbol{y}\succeq 0}\mathcal{L}(\boldsymbol{x}(m),\boldsymbol{y},\boldsymbol{\omega})$$
$$=\min_{\boldsymbol{y}}\Big\{\mathcal{F}(\boldsymbol{x}(m),\boldsymbol{\omega})+\sum\nolimits_{i=1}^{N}\Big[\sum\nolimits_{l=1}^{L}(\tau_{il}-\lambda_{il})-1\Big]o_i$$
$$+\sum\nolimits_{i=1,i\neq j}^{N}\sum\nolimits_{j=1}^{N}\Big[\sum\nolimits_{k=1}^{M}(\alpha_{ij}^k-\beta_{ij}^k)+\gamma_{ij}-\delta_{ij}\Big](t_i^s-t_j^s)$$
$$+\sum\nolimits_{i=1}^{N}\varepsilon_i t_i^s+\sum\nolimits_{i=1,i\neq j}^{N}\sum\nolimits_{l=1}^{L}\frac{\sum_{j=1}^{N}\sum_{k=1}^{M}(\alpha_{ij}^k+\beta_{ij}^k)}{f_l}h_{il}$$
$$+\sum\nolimits_{i=1,i\neq j}^{N}\sum\nolimits_{l=1}^{L}\frac{\sum_{j=1}^{N}(\gamma_{ij}-\delta_{ij})p_{ij}}{f_l}h_{il}$$
$$+\sum\nolimits_{i=1}^{N}\sum\nolimits_{l=1}^{L}\Big(\frac{\varepsilon_i+\eta}{f_l}+\theta_{il}+\lambda_{il}-\tau_{il}\Big)h_{il}\Big\}.$$

For the dual function $\mathcal{R}(\boldsymbol{x}(m),\boldsymbol{\omega})$, since the continuous variables o_i, t_i^s and h_{il} are positive, they are finite only when following constraints hold:

$$\sum\nolimits_{l=1}^{L}(\tau_{il}-\lambda_{il})-1\geq 0,\ 1\leq i\leq N, \tag{23}$$

$$\sum\nolimits_{k=1}^{M}(\alpha_{ij}^k-\beta_{ij}^k)+\gamma_{ij}-\delta_{ij}\geq 0,\ 1\leq i,j\leq N, i\neq j, \tag{24}$$

$$\frac{\sum_{j=1}^{N}\big[\sum_{k=1}^{M}(\alpha_{ij}^k+\beta_{ij}^k)+(\gamma_{ij}-\delta_{ij})p_{ij}\big]}{f_l}\geq 0,$$
$$1\leq i\leq N, i\neq j, 1\leq l\leq L, \tag{25}$$

$$\frac{\varepsilon_i+\eta}{f_l}+\theta_{il}+\lambda_{il}-\tau_{il}\geq 0,\ 1\leq i\leq N, 1\leq l\leq L. \tag{26}$$

Thus, the dual problem associated with the SP (22) is

$$\textbf{DSP}:\max_{\boldsymbol{\omega}\succeq\boldsymbol{0}}\ \mathcal{F}(\boldsymbol{x}(m),\boldsymbol{\omega}) \tag{27}$$
$$\text{s.t.}\ \ (23),(24),(25),(26).$$

For convenience, the matrices and the vectors are used to denote the constraints and the variables. Hence, the PP1 (20) is reformulated as

$$\textbf{PP1}:\min_{\boldsymbol{x},\boldsymbol{y}}\ \Phi(\boldsymbol{x},\boldsymbol{y})=\boldsymbol{f}'\boldsymbol{y} \tag{28}$$
$$\text{s.t.}\ \begin{cases}\boldsymbol{A}\boldsymbol{x}\preceq\boldsymbol{b}_1,\\ \boldsymbol{C}\boldsymbol{x}+\boldsymbol{D}\boldsymbol{y}\preceq\boldsymbol{b}_2.\end{cases}$$

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

Therefore, the SP and the DSP associated with the PP1 (28) are expressed as

$$\textbf{SP}:\min_{\boldsymbol{y}\succeq 0}\ \Phi(\boldsymbol{x}(m),\boldsymbol{y})=\boldsymbol{f}'\boldsymbol{y} \tag{29}$$
$$\text{s.t.}\ \ \boldsymbol{C}\boldsymbol{x}(m)+\boldsymbol{D}\boldsymbol{y}\preceq\boldsymbol{b}_2.$$

$$\textbf{DSP}:\max_{\boldsymbol{\omega}\succeq 0}\ \mathcal{F}(\boldsymbol{x}(m),\boldsymbol{\omega})=(\boldsymbol{C}\boldsymbol{x}(m)-\boldsymbol{b}_2)'\boldsymbol{\omega} \tag{30}$$
$$\text{s.t.}\ \ \boldsymbol{f}+\boldsymbol{D}'\boldsymbol{\omega}\succeq 0.$$

Since DSP (30) is an LP problem, it can be efficiently solved using polynomial-time algorithms, such as simplex method or interior point method [4].

Step 3 – Solving Master Problem: Based on the solution of the DSP (30), two different types of constraints are generated and are added into the MP (21).

a) If the DSP is infeasible: the SP (29) has an unbounded solution. Hence, the PP1 (28) has no feasible solution.

b) If the DSP has a bounded solution $\boldsymbol{\omega}(m)$: the SP (29) is feasible due to the strong duality, and $\mathcal{A}\leftarrow\{m\}\cup\mathcal{A}$. Denote $(\hat{\Phi}(m),\boldsymbol{x}(m))$ and $\boldsymbol{y}(m)$ as the solutions of the MP and the SP at the m^{th} iteration, respectively. Since i) $\boldsymbol{x}(m)$ is a feasible solution (not optimal) of the PP1 (28), and ii) $\boldsymbol{f}'\boldsymbol{y}(m)=(\boldsymbol{C}\boldsymbol{x}(m)-\boldsymbol{b}_2)'\boldsymbol{\omega}(m)$ due to the strong duality, the upper bound of Φ^* at the m^{th} iteration is updated by $\Phi_u(m)=\min\{\Phi_u(m-1),(\boldsymbol{C}\boldsymbol{x}(m)-\boldsymbol{b}_2)'\boldsymbol{\omega}(m)\}$. Since $\hat{\Phi}(m)$ and $(\boldsymbol{C}\boldsymbol{x}(m)-\boldsymbol{b}_2)'\boldsymbol{\omega}(m)$ are the lower bound and the upper bound of Φ^* at the m^{th} iteration, respectively, we have $\hat{\Phi}(m)<(\boldsymbol{C}\boldsymbol{x}(m)-\boldsymbol{b}_2)'\boldsymbol{\omega}(m)$. To avoid selecting the non-optimal solution $\boldsymbol{x}(m)$ again at the next iteration, a new feasibility constraint

$$\hat{\Phi}\geq(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)'\boldsymbol{\omega}(m)=\mathcal{F}(\boldsymbol{x},\boldsymbol{\omega}(m)) \tag{31}$$

is generated and added into the MP at the $(m+1)^{th}$ iteration.

Note that i) the objective functions of the SP (29) and the PP1 (28) are the same, and ii) the SP (29) and the DSP (30) are equivalent due to the strong duality. From (31), we can see that the auxiliary variable $\hat{\Phi}$ has the same physical meaning as the objective function of the PP1 (28).

c) If the DSP has an unbounded solution: i.e., $\mathcal{F}(\boldsymbol{x}(m),\boldsymbol{\omega}(m))=+\infty$, due to the strong duality, the SP (29) has no feasible solution with $\boldsymbol{x}(m)$, 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 is 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)

$$\textbf{FCP}:\min_{\boldsymbol{\xi}\succeq 0}\ \boldsymbol{1}'\boldsymbol{\xi} \tag{32}$$
$$\text{s.t.}\ \ \boldsymbol{C}\boldsymbol{x}(m)+\boldsymbol{D}\boldsymbol{y}\preceq\boldsymbol{b}_2+\boldsymbol{\xi}.$$

To develop the dual problem of FCP, we introduce the positive Lagrange multipliers $\boldsymbol{\nu}(m)=[\nu_i(m)]$ $(1\leq i\leq v)$ to the FCP. Hence, the dual problem associated with the FCP (DFCP) is

$$\textbf{DFCP}:\max_{\boldsymbol{\nu}\succeq 0}\ \mathcal{F}(\boldsymbol{x}(m),\boldsymbol{\nu})=(\boldsymbol{C}\boldsymbol{x}(m)-\boldsymbol{b}_2)'\boldsymbol{\nu} \tag{33}$$
$$\text{s.t.}\ \ \boldsymbol{1}-\boldsymbol{\nu}\succeq 0.$$

Since FCP and DFCP are LP problems, they can be solved by methods used to solve the DSP. Denote $\boldsymbol{\xi}(m)$ and $\boldsymbol{\nu}(m)$ as the solutions of FCP and DFCP at the m^{th} iteration, respectively. If the SP (29) exists infeasible constraints, the related relax variables are non-zero, while the others are zero. Hence, we have $\boldsymbol{1}'\boldsymbol{\xi}(m)>0$. Since the strong duality is guaranteed between the FCP and its dual problem, we get $\boldsymbol{1}'\boldsymbol{\xi}(m)=(\boldsymbol{C}\boldsymbol{x}(m)-\boldsymbol{b}_2)'\boldsymbol{\nu}(m)>0$. To avoid selecting the infeasible solution $\boldsymbol{x}(m)$ again, a new infeasibility constraint

$$0\geq(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)'\boldsymbol{\nu}(m)=\mathcal{F}(\boldsymbol{x},\boldsymbol{\nu}(m)) \tag{34}$$

is generated and added into the MP at the $(m+1)^{th}$ iteration.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

Note that in constraints (31) and (34), all the parameters are constant except $\hat{\Phi}$ and $\boldsymbol{x}$, which are the MP variables at the $(m+1)^{th}$ iteration. The reason we solve DFCB rather than FCB is that the objective function of DFCB (i.e., $\mathcal{F}(\boldsymbol{x},\boldsymbol{\nu})$) is a function with respect to the variables $\boldsymbol{x}$, but not $\mathbf{1}'\boldsymbol{\xi}$ (i.e., the objective function of the FCB). In other word, $0 \geq \mathbf{1}'\boldsymbol{\xi}(m)$ is a less useful constraint for the MP. With this new added feasibility/infeasibility constraint, the MP is solved again to obtain a new solution $\boldsymbol{x}(m+1)$ for the next round iteration. The iteration stops when $|\Phi_u(m) - \Phi_l(m)| \leq \epsilon$ is satisfied.

C. Convergence Analysis

Although the MP (21) is composed of binary variables $\boldsymbol{x}$ and of a continuous variable $\hat{\Phi}$, it can be solved by only considering the binary variables [27], [29]. The reason is as follow. From (21), we can see that the real constraints are (2), (3) and $0 \geq \mathcal{F}(\boldsymbol{x},\boldsymbol{\nu}(\vartheta))$ $(\forall \varsigma \in \mathcal{A})$. Items $\mathcal{F}(\boldsymbol{x},\boldsymbol{\omega}(\varsigma))$ $(\forall \vartheta \in \mathcal{B})$ can be viewed as the objective functions. Comparing the following ILP problem

$$\mathbf{MP}_i : \hat{\Phi}_r(i) = \min_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{x},\boldsymbol{\omega}(i)) \quad \text{s.t.} \begin{cases} (2),(3), \\ 0 \geq \mathcal{F}(\boldsymbol{x},\boldsymbol{\nu}(i)),\ \forall i \in \mathcal{B}, \\ c_{il}, q_{ik} \in \{0,1\},\ \forall i \in \mathcal{N},\ \forall l \in \mathcal{L}. \end{cases} \tag{35}$$

with the MP, since the constraints of the MP_i $(\forall i \in \mathcal{A})$ and the MP are the same, we have $\hat{\Phi}(m) = \max_{\forall i \in \mathcal{A}}\{\hat{\Phi}_r(i)\}$. Hence, the MP is an ILP, which is an $\mathcal{NP}$-complete problem [30].

Lemma 3.2: The lower bound $\Phi_l(m)$ (upper bound $\Phi_u(m)$) on the optimal objective function value Φ^* is derived from the solution of the MP (21) (SP (22)) at the m^{th} iteration.

Proof: First, we prove that $\hat{\Phi}(m)$ is a lower bound of Φ^*, where $\hat{\Phi}(m)$ is the solution of the MP at the m^{th} iteration. Without loss of generality, we assume that $\hat{\Phi}(m) = \hat{\Phi}_r(\rho) = \max_{\forall i \in \mathcal{A}}\{\hat{\Phi}_r(i)\}$, where $\rho \in \mathcal{A}$. Hence, we get

$$\begin{aligned} \hat{\Phi}(m) &= \min_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{x},\boldsymbol{\omega}(\rho)) \\ &\leq \mathcal{F}(\boldsymbol{x}^*,\boldsymbol{\omega}(\rho)) \qquad (36a) \\ &\leq \max_{\boldsymbol{\omega} \succeq 0} \mathcal{F}(\boldsymbol{x}^*,\boldsymbol{\omega}) = \Phi^*, \qquad (36b) \end{aligned}$$

where $\min_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{x},\boldsymbol{\omega}(\rho))$ and $\max_{\boldsymbol{\omega} \succeq 0} \mathcal{F}(\boldsymbol{x}^*,\boldsymbol{\omega})$ are the optimal objective function values of the MP_ρ (35) and the DSP (30), respectively. (36a) holds since $\boldsymbol{x}^*$ is not the optimal solution of MP_ρ, and (36b) holds since by solving the DSP with the optimal binary variables $\boldsymbol{x}^*$ we can find the optimal objective function value Φ^*. (36) shows that $\Phi_l(m) = \hat{\Phi}(m)$ is a lower bound of Φ^*.

Next, we prove that $\Phi_u(m) = \min\{\Phi_u(m-1), \mathcal{F}(\boldsymbol{x}(m), \boldsymbol{\omega}(m))\}$ is an upper bound of Φ^*. Note that $\min\{\Phi_u(m-1), \mathcal{F}(\boldsymbol{x}(m),\boldsymbol{\omega}(m))\} = \min_{1 \leq i \leq m}\{\mathcal{F}(\boldsymbol{x}(i),\boldsymbol{\omega}(i))\}$, where $\boldsymbol{x}(i)$ and $\boldsymbol{\omega}(i)$ are the solutions of the MP (21) and the DSP (30) at the i^{th} iteration, respectively. Depending on the solution of the DSP, $\mathcal{F}(\boldsymbol{x}(i),\boldsymbol{\omega}(i))$ can be either finite or infinite. If the DSP has an unbounded solution (i.e., $\mathcal{F}(\boldsymbol{x}(i),\boldsymbol{\omega}(i)) = +\infty$), it is obvious that $+\infty$ is an upper bound of Φ^*. Thus, we focus on the case when the DSP has a bounded solution $(\boldsymbol{x}(i),\boldsymbol{\omega}(i))$. Due to the duality between the SP and the DSP, we have

$$\mathcal{F}(\boldsymbol{x}(i),\boldsymbol{\omega}(i)) = \min_{\boldsymbol{y} \succeq 0} \Phi(\boldsymbol{x}(i),\boldsymbol{y}) \geq \min_{\boldsymbol{y} \succeq 0} \Phi(\boldsymbol{x}^*,\boldsymbol{y}) = \Phi^*, \tag{37}$$

where $\min_{\boldsymbol{y} \succeq 0} \Phi(\boldsymbol{x},\boldsymbol{y})$ is the optimal objective function value of the SP under the given binary variables $\boldsymbol{x}$. From (37), we can see that $\Phi_u(m) = \min_{1 \leq i \leq m}\{\mathcal{F}(\boldsymbol{x}(i),\boldsymbol{\omega}(i))\}$ is an upper bound of Φ^*. ■

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

Proof: Note that i) the aim of the MP is to minimize the objective function, ii) the optimal objective function value of the MP equals to the lower bound (i.e., $\Phi_l(m) = \hat{\Phi}(m)$), iii) the optimal objective function values of the MP at previous m iterations (i.e., $\{\hat{\Phi}(0),\ldots,\hat{\Phi}(m)\}$) have been excluded by the constraints (31) and (34), and iv) with iteration number m increasing, more constraints are added into the MP (i.e., the feasible region of the MP shrinks). The lower bound at the $(m+1)^{th}$ iteration (i.e., $\Phi_l(m+1)$) is larger than the previous lower bounds $\{\Phi_l(0),\ldots,\Phi_l(m)\}$. On the other hand, since $\Phi_u(m)$ is achieved by $\Phi_u(m) = \min\{\Phi_u(m-1), (\boldsymbol{C}\boldsymbol{x}(m) - \boldsymbol{b}_2)'\boldsymbol{\omega}(m)\}$, the upper bound at the $(m+1)^{th}$ iteration (i.e., $\Phi_u(m+1)$) is not larger than the previous upper bounds $\{\Phi_u(0),\ldots,\Phi_u(m)\}$. ■

Theorem 3.1: At each iteration with a new feasibility constraint (31) or infeasibility constraint (34) added into the MP (21), the solution obtained by JDQT converges to the global optimal value within a finite number of iterations.

Proof: At each iteration m, by solving the MP and the SP, we obtain a lower bound $\Phi_l(m)$ and an upper bound $\Phi_u(m)$ of the optimal objective function value Φ^*. The bound sequence $\{\Phi_l(0),\ldots,\Phi_l(m)\}$ is increasing, while the upper bound sequence $\{\Phi_u(0),\ldots,\Phi_u(m)\}$ is decreasing. In addition, at each iteration, there is always one new constraint (feasibility constraint (31) or infeasibility constraint (34)) added into the MP to exclude these non-optimal or infeasible values of binary variables $\boldsymbol{x}$. Since the dimension of binary variables $\boldsymbol{x}$ is finite, according to the *Theorem 2.4* in [31], the solution converges to the global optimal value within a finite number of iterations. ■

D. Accelerated Solution Method: AJDQT

Although the solution provided by JDQT is optimal, this method cannot be used to efficiently solve large problem sizes. This is due to the following reasons: i) as the MP (21) is an ILP, this problem is still hard to solve directly, compared with the SP, and ii) at each iteration, a new feasibility constraint (31) or infeasibility constraint (34) is added into the MP. With an increasing number of iterations, the computational complexity and the size of MP both increase. In order to circumvent these difficulties, we propose an accelerated JDQT (AJDQT) algorithm to further reduce the computational complexity of JDQT. This algorithm contains two parts and its structure is shown in Fig. 4.

1) Relaxation of The MP: The computational complexity of JDQT is dominated by the cost of solving the MP at each iteration. In order to reduce the computing time of JDQT, we relax the binary variables $\boldsymbol{x}$ to be continuous variables with their ranges in $[0,1]$. Hence, the relaxed MP is formulated as

$$\mathbf{MP1} : \min_{\hat{\Phi},\boldsymbol{x}} \hat{\Phi} \quad \text{s.t.} \begin{cases} (2),(3), \\ \hat{\Phi} \geq \mathcal{F}(\boldsymbol{x},\boldsymbol{\omega}(\varsigma)),\ \forall \varsigma \in \mathcal{A}, \\ 0 \geq \mathcal{F}(\boldsymbol{x},\boldsymbol{\nu}(\vartheta)),\ \forall \vartheta \in \mathcal{B}, \\ 0 \leq c_{il}, q_{ik} \leq 1, \hat{\Phi} \geq 0,\ \forall i \in \mathcal{N},\ \forall l \in \mathcal{L}. \end{cases} \tag{38}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

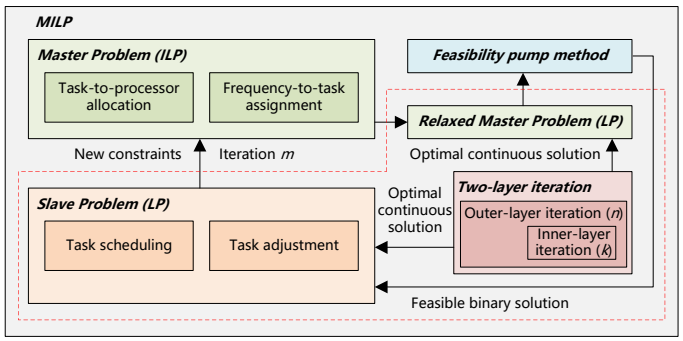

Fig. 4. Structure of the Accelerated JDQT (AJDQT) algorithm.

Compared with MP, MP1 is an LP problem since all the variables are continuous. Note that if the DSP (30) is solved with $\boldsymbol{x}(m)$, where $\boldsymbol{x}(m)$ is the solution of the MP1, the DSP may be infeasible since $\boldsymbol{x}(m)$ may not be a binary solution. One way to solve this problem is to round the solution of MP1 (i.e., $\boldsymbol{x}(m)$) to the nearest binary solution that is feasible to the MP. Such a binary solution, denoted by $\overline{\boldsymbol{x}}(m)$, can be found by heuristics such as Feasibility Pump (FP) method [32].

Theorem 3.2: Assume that $\overline{\boldsymbol{x}}(m)$ is an arbitrary feasible solution of the MP. The feasibility and infeasibility constraints generated by solving the DSP (30) with $\overline{\boldsymbol{x}}(m)$ do not exclude the optimal solution of PP1 (20).

Proof: If the DSP has a bounded solution $\overline{\boldsymbol{\omega}}(m)$ with $\overline{\boldsymbol{x}}(m)$, the feasibility constraint is

$$\hat{\Phi} \geq \mathcal{F}(\boldsymbol{x}, \overline{\boldsymbol{\omega}}(m)). \tag{39}$$

On the other hand, if the DSP has an unbounded solution with $\overline{\boldsymbol{x}}(m)$, the infeasibility constraint is

$$0 \geq \mathcal{F}(\boldsymbol{x}, \overline{\boldsymbol{\nu}}(m)). \tag{40}$$

Note that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is the optimal solution of PP1, and Φ^* is the optimal objective function value. In the following, we prove that the optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ does not violate the constraints (39) and (40).

If the DSP has a bounded solution $\overline{\boldsymbol{\omega}}(m)$ with $\overline{\boldsymbol{x}}(m)$, suppose that $\boldsymbol{x}^*$ and Φ^* violate the feasibility constraint (i.e., $\Phi^* < \mathcal{F}(\boldsymbol{x}^*, \overline{\boldsymbol{\omega}}(m))$). This contradicts the fact that Φ^* is the optimal objective function value of the DSP with $\boldsymbol{x}^*$ (i.e., $\Phi^* = \max_{\boldsymbol{\omega} \succeq 0} \mathcal{F}(\boldsymbol{x}^*, \boldsymbol{\omega}) \geq \mathcal{F}(\boldsymbol{x}^*, \overline{\boldsymbol{\omega}}(m))$). Hence, the feasibility constraint (39) will not exclude the optimal solution $\boldsymbol{x}^*$. On the other hand, if the DSP has an unbounded solution with $\overline{\boldsymbol{x}}(m)$, this solution will be excluded by the infeasibility constraint (40). Since $\boldsymbol{x}^* \neq \overline{\boldsymbol{x}}(m)$, $\boldsymbol{x}^*$ does not violate the infeasibility constraint (40). ■

2) Distributed Solution: Two-layer Subgradient-based Algorithm: Based on the structure of the MP1 (38) and the SP (29), we design a distributed solution based on two-layer subgradient algorithm to solve these problems. For convenience, the previous problem formulations of MP1 and SP are reformulated in an abstract manner as follows:

$$\min_{\boldsymbol{z}} \ \boldsymbol{g}'\boldsymbol{z} \tag{41}$$
$$\text{s.t.} \ \begin{cases} \boldsymbol{b}_i'\boldsymbol{z} \leq e_i, \ 1 \leq i \leq p, \\ \underline{z}_j \leq z_j \leq \overline{z}_j, \ 1 \leq j \leq q, \end{cases}$$

where $\boldsymbol{z}$ is a vector of continuous variables, $\boldsymbol{g}$ is a vector of objective function coefficients, $\boldsymbol{B} \triangleq [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_p]'$ and $\boldsymbol{e} \triangleq [e_1, \ldots, e_p]'$ are the constraints related matrix and vector, respectively.

Lemma 3.4: The MP1 (38) and the SP (29) are convex.

Proof: Let $\mathcal{R}(\boldsymbol{z}) \triangleq \boldsymbol{g}'\boldsymbol{z}$ and $\mathcal{G}_i(\boldsymbol{z}) \triangleq \boldsymbol{b}_i'\boldsymbol{z} - e_i$ $(1 \leq i \leq p)$. Since

$$\mathcal{O} = \begin{bmatrix} \frac{\partial^2 \mathcal{R}(\boldsymbol{z})}{\partial z_1^2} & \cdots & \frac{\partial^2 \mathcal{R}(\boldsymbol{z})}{\partial z_1 \partial z_q} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{R}(\boldsymbol{z})}{\partial z_q \partial z_1} & \cdots & \frac{\partial^2 \mathcal{R}(\boldsymbol{z})}{\partial z_q^2} \end{bmatrix} = \boldsymbol{0}_{q \times q},$$

$$\mathcal{C}_i = \begin{bmatrix} \frac{\partial^2 \mathcal{G}_i(\boldsymbol{z})}{\partial z_1^2} & \cdots & \frac{\partial^2 \mathcal{G}_i(\boldsymbol{z})}{\partial z_1 \partial z_q} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{G}_i(\boldsymbol{z})}{\partial z_q \partial z_1} & \cdots & \frac{\partial^2 \mathcal{G}_i(\boldsymbol{z})}{\partial z_q^2} \end{bmatrix} = \boldsymbol{0}_{q \times q}, \ 1 \leq i \leq p,$$

the Hessian matrices of functions $\mathcal{R}(\boldsymbol{z})$ and $\mathcal{G}_i(\boldsymbol{z})$ are positive semi-definite. Thus, the objective function and the constraints of the problem (41) are convex. According to the definition of convex problem [28], the MP1 and the SP are convex. ■

By introducing the positive Lagrange multipliers $\boldsymbol{\psi} \triangleq [\psi_1, \ldots, \psi_p]'$ to the problem (41), the corresponding Lagrangian is $\mathcal{H}(\boldsymbol{z}, \boldsymbol{\psi}) = \boldsymbol{g}'\boldsymbol{z} + \boldsymbol{\psi}'(\boldsymbol{B}\boldsymbol{z} - \boldsymbol{e})$. Hence, the dual function is $\mathcal{D}(\boldsymbol{\psi}) = \min_{\underline{\boldsymbol{z}} \preceq \boldsymbol{z} \preceq \overline{\boldsymbol{z}}} \mathcal{H}(\boldsymbol{z}, \boldsymbol{\psi})$, and the dual problem associated with problem (41) is

$$\max_{\boldsymbol{\psi} \succeq 0} \ \mathcal{D}(\boldsymbol{\psi}) = -\boldsymbol{\psi}'\boldsymbol{e} \tag{42}$$
$$\text{s.t.} \ \ \boldsymbol{g} + \boldsymbol{B}'\boldsymbol{\psi} \succeq 0.$$

For simplicity and generality, we introduce indexes n and k to count the outer-layer and inner-layer iterations, respectively. The inner-layer iteration aims to update variables $\boldsymbol{z}$ under the given Lagrange multipliers $\boldsymbol{\psi}$, while the outer-layer iteration aims to update Lagrange multipliers $\boldsymbol{\psi}$ under the given variables $\boldsymbol{z}$. The details are as follows.

a) Inner-Layer Iteration: Assume that the current outer-layer iteration is n. Based on the updated result of previous outer-layer iteration (i.e., $\boldsymbol{\psi}(n)$), the variable z_i is iteratively updated by

$$z_i(n, k+1) = \left[z_i(n,k) - \sigma \frac{\partial \mathcal{H}(\boldsymbol{z}(n), \boldsymbol{\psi}(n))}{\partial z_i(n)} \right]_{\underline{z}_i}^{\overline{z}_i}, 1 \leq i \leq q, \tag{43}$$

where $[z_i]_{\underline{z}_i}^{\overline{z}_i} = \max\{\underline{z}_i, \min\{z_i, \overline{z}_i\}\}$ and σ is a positive small step-size.

Since i) the functions $\mathcal{R}(\boldsymbol{z})$ and $\mathcal{G}_i(\boldsymbol{z})$ are convex, and ii) the sum operation preserves the convexity, $\mathcal{H}(\boldsymbol{z}, \boldsymbol{\psi})$ is a convex function. Denote $\zeta \geq 0$ as a small tolerance. The inner-layer iteration can start from an arbitrary initial point $\boldsymbol{z}(n, 0)$, and stops when $||\boldsymbol{z}(n, k+1) - \boldsymbol{z}(n, k)||_2 \leq \zeta$. Hence, by iterating (43), we can obtain $\boldsymbol{z}(n)$ (i.e., the optimal solution of dual function $\mathcal{D}(\boldsymbol{\psi})$ under the given $\boldsymbol{\psi}(n)$).

b) Outer-Layer Iteration: Based on the updated results of the previous inner-layer and outer-layer iterations (i.e., $\boldsymbol{z}(n)$ and $\boldsymbol{\psi}(n)$), the Lagrange multiplier ψ_j is iteratively updated by

$$\psi_j(n+1) = \left[\psi_j(n) + \sigma \frac{\partial \mathcal{D}(\boldsymbol{\psi}(n))}{\partial \psi_j(n)} \right]^+, 1 \leq j \leq p, \tag{44}$$

where $[\psi_j]^+ = \max\{0, \psi_j\}$.

Since the Lagrangian $\mathcal{H}(\boldsymbol{z}, \boldsymbol{\psi})$ is convex, its dual function $\mathcal{D}(\boldsymbol{\psi})$ is concave [28]. Hence, the outer-layer iteration can start from the arbitrary initial points $\boldsymbol{\psi}(0)$, and stops when $||\boldsymbol{\psi}(n+1) - \boldsymbol{\psi}(n)||_2 \leq \zeta$. Note that $\mathcal{H}(\boldsymbol{z}, \boldsymbol{\psi})$ and $\mathcal{D}(\boldsymbol{\psi})$ are the linear functions with respect to the variables $\boldsymbol{z}$ and the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

Lagrange multipliers $\boldsymbol{\psi}$, respectively. The partial derivatives $\frac{\partial \mathcal{H}(\boldsymbol{z}(n),\boldsymbol{\psi}(n))}{\partial z_i(n)}$ and $\frac{\mathcal{D}(\boldsymbol{\psi}(n))}{\partial \psi_j(n)}$ are coefficients, which implies that the variable $z_i(n,k)$ and the Lagrange multiplier $\psi_j(n)$ are updated individually (i.e., the SP (29) and the MP1 (38) can be solved in a distributed manner).

c) Convergence Analysis: Note that problem (41) is convex, there is no duality gap between this problem and its dual problem (42) (i.e., solving the dual problem equals to solving its primal problem). The following *Theorem 3.3* proves the convergence of the algorithm.

Definition 3.2: (*Statistical convergence [33]*) For an optimization problem $\min_{\boldsymbol{\chi}} \mathcal{S}(\boldsymbol{\chi})$ and an iterative solution with step size σ, let $\boldsymbol{\chi}^*$ and $\bar{\boldsymbol{\chi}} = \frac{1}{n}\sum_{l=1}^{n}\boldsymbol{\chi}(l)$ be the optimal solution and the average solution found by the n^{th} iteration, respectively. The statistical convergence means that for any $\phi \geq 0$, there exists an σ such that $\limsup_{n\to\infty}(\mathcal{S}(\bar{\boldsymbol{\chi}})-\mathcal{S}(\boldsymbol{\chi}^*)) \leq \phi$.

Theorem 3.3: With the iterations between the outer-layer and the inner-layer, the Lagrange multipliers $\boldsymbol{\psi}$ statistically converge to the optimal values $\boldsymbol{\psi}^*$ when step size σ is a small enough value.

Proof: We construct a Lyapunov function $\mathcal{V}(\boldsymbol{\psi}(n)) = \sum_{j=1}^{p}(\psi_j^* - \psi_j(n))^2$, where ψ_j^* is the optimal value of ψ_j. Denote $\mathcal{M}_j(n) \triangleq \frac{\mathcal{D}(\boldsymbol{\psi}(n))}{\partial \psi_j(n)}$ as the sub-gradient of the dual function $\mathcal{D}(\boldsymbol{\psi}(n))$ at the point $\psi_j(n)$. According to $\psi_j(n+1) = [\psi_j(n) + \sigma\mathcal{M}_j(n)]^+$, we have $\psi_j(n+1) \geq 0$ and $\psi_j(n) \geq 0$. Since $\sigma > 0$ and $\mathcal{M}_j(n)$ could be either positive or negative, we get $\psi_j(n+1) \geq \psi_j(n) + \sigma\mathcal{M}_j(n)$. And further, $(\psi_j^* - \psi_j(n+1))^2 \leq (\psi_j^* - \psi_j(n) - \delta\mathcal{M}_j(n))^2$ due to $\psi_j^* \geq 0$. Hence, we have

$$
\begin{aligned}
&\mathcal{V}(\boldsymbol{\psi}(n+1)) = \sum\nolimits_{j=1}^{p}(\psi_j^* - \psi_j(n+1))^2 \\
\leq &\sum\nolimits_{j=1}^{p}(\psi_j^* - \psi_j(n) - \sigma\mathcal{M}_j(n))^2 \\
= &\mathcal{V}(\boldsymbol{\psi}(n)) + \sum\nolimits_{j=1}^{p}[2\sigma(\psi_j(n) - \psi_j^*)\mathcal{M}_j(n) + \sigma^2\mathcal{M}_j(n)^2] \\
\leq &\mathcal{V}(\boldsymbol{\psi}(n)) + 2\sigma[\mathcal{D}(\boldsymbol{\psi}(n)) - \mathcal{D}(\boldsymbol{\psi}^*)] + \sum\nolimits_{j=1}^{p}\sigma^2\mathcal{M}_j(n)^2 \\
\leq &\mathcal{V}(\boldsymbol{\psi}(1)) + \sum\nolimits_{l=1}^{n}\Big\{2\sigma[\mathcal{D}(\boldsymbol{\psi}(l)) - \mathcal{D}(\boldsymbol{\psi}^*)] \\
&+ \sum\nolimits_{j=1}^{p}\sigma^2\mathcal{M}_j(l)^2\Big\}, \qquad (45)
\end{aligned}
$$

where the second inequality holds since $\mathcal{D}(\boldsymbol{\psi})$ is a concave function, and, thus, based on the definition of subgradient we have $\sum_{j=1}^{p}(\psi_j(n) - \psi_j^*)\mathcal{M}_j(n) \leq \mathcal{D}(\boldsymbol{\psi}(n)) - \mathcal{D}(\boldsymbol{\psi}^*)$.

Let $\overline{\boldsymbol{\psi}}(n) = \frac{1}{n}\sum_{l=1}^{n}\boldsymbol{\psi}(l)$. Based on the concavity of dual function $\mathcal{D}(\boldsymbol{\psi})$ and Jensen's inequality [28], we get

$$
\sum\nolimits_{l=1}^{n}[\mathcal{D}(\boldsymbol{\psi}(l)) - \mathcal{D}(\boldsymbol{\psi}^*)] \leq n[\mathcal{D}(\overline{\boldsymbol{\psi}}(n)) - \mathcal{D}(\boldsymbol{\psi}^*)]. \qquad (46)
$$

Assume that $\sum_{j=1}^{p}\psi_j(l)^2 \leq \mathcal{B}$ since $\{\psi_j(l)\}$ are bounded. Substituting (46) into (45) and noting that $\mathcal{V}(\boldsymbol{\psi}(n+1)) \geq 0$, we have $\mathcal{D}(\boldsymbol{\psi}^*) - \mathcal{D}(\overline{\boldsymbol{\psi}}(n)) \leq \frac{\mathcal{V}(\boldsymbol{\psi}(1)) + n\sigma^2\mathcal{B}}{2n\sigma}$. From $\limsup_{n\to\infty}(\mathcal{D}(\boldsymbol{\psi}^*)) - \mathcal{D}(\overline{\boldsymbol{\psi}}(n)) \leq \frac{\sigma\mathcal{B}}{2}$, we can see that the Lagrange multipliers $\boldsymbol{\psi}$ statistically converge to the optimal values $\boldsymbol{\psi}^*$ when step size σ is a small enough value. ■

Hence, by iterating the equations (43) and (44), we obtain the optimal solutions of the MP1 (38) and the SP (29).

3) Run-Time Complexity: The MP1 (38) and the SP (29) can be solved by either the interior point method or the proposed distributed method. Using the interior point method, the computational complexity is $O(d^3)$, where d is the number of the variables [34]. Based on the structures of MP1 and SP, we have $d = N(2+L)$ and $d = 2NL$, respectively. When using the proposed distributed method, since the inter-layer and the outer-layer iterations are based on the subgradient method, the computational complexity is $O(U^2R^2\zeta^{-2})$ [35]. U is the distance between an optimal solution and the initial point, and R is a Lipschitz constant for the objective function. Therefore, the proposed distributed method has lower computational complexity.

IV. Simulation Results

The multicore platform model used in the experiments is based on 70 nm technology [5]. The processor operates at five voltage levels in the range of $[0.65\ V, 0.85\ V]$ with a step of 50 mV (i.e., $L = 5$). The power of the processor in the idle state is set to $P_0^s = 80\ \mu W$, while the corresponding frequency f_l, the dynamic power P_l^d, and the static power P_l^s under different voltage levels are shown in Table I. The number of processors (i.e., M) is tuned from 4 to 8 with a step of 2. We generate task graphs with 10 to 50 tasks (i.e., the task number N is tuned from 10 to 50 with a step of 5). The WCECs of the mandatory part M_i and the maximum optional part O_i of task τ_i ($\forall i \in \mathcal{N}$) are assumed to be in the range $[4\times10^7, 6\times10^8]$ [17]. To set the absolute deadline of task τ_i, we introduce temporary start time $\hat{t}_s^i$ and relative deadline r_i for task τ_i. Then, the absolute deadline d_i is calculated by $d_i = \hat{t}_s^i + r_i$. The relative deadline r_i is assumed to be in the range $[\underline{r}_i, \overline{r}_i]$, where $\underline{r}_i = \min_{\forall l\in\mathcal{L}}\{\frac{M_i+O_i}{f_l}\}$ and $\overline{r}_i = \max_{\forall l\in\mathcal{L}}\{\frac{M_i+O_i}{f_l}\}$ are the minimum time and the maximum time required to execute $\{M_1+O_1,\ldots,M_N+O_N\}$ cycles, respectively. We calculate the value of temporary start time $\hat{t}_s^i$ through the EOD matrix $\boldsymbol{P}$. When $p_{ij} = 1$ (i.e., task τ_i precedes task τ_j and τ_j is the closest task of τ_i), we set the temporary end time of task τ_i equal to the temporary start time of task τ_j (i.e., $\hat{t}_e^i = \hat{t}_s^i + r_i = \hat{t}_s^j$). If τ_i is the first task in one hyper-period, we set $\hat{t}_s^i = 0$. Moreover, we assume that the hyper-period of the tasks is $H = \max_{\forall i\in\mathcal{N}}\{d_i\}$. Since system is in the medium energy state, the energy supply is set to $E_s = \eta E_h$, where $E_h = MHP_0^s + \sum_{i=1}^{N}[\min_{\forall l\in\mathcal{L}}\frac{M_i+O_i}{f_l}(P_l^s + P_l^d - P_0^s)]$ is the minimum energy required to execute $\{M_1+O_1,\ldots,M_N+O_N\}$ cycles. The energy efficiency factor η is tuned from 0.8 to 0.9 with a step of 0.05. Note that different multi-core platforms and task graphs will lead to different parameters $\{\boldsymbol{A},\boldsymbol{C},\boldsymbol{D},\boldsymbol{f},\boldsymbol{b}_1,\boldsymbol{b}_2\}$ for the PP1 (28). However, the problem structures under different parameters are the same, and, thus, different algorithms can be compared under given system parameters.

TABLE I. Dynamic power consumption and static power consumption for 70 nm processor

$v_l\ (V)$	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

Initially, we compare the behavior (system QoS and computing time) of the proposed JDQT with: i) optimal approaches, i.e. Branch and Bound method (B&B) [36], [37] and Branch and Cut method (B&C) [38], which are known to provide optimal solution for the MILP problem – as far as we know no optimal algorithm exists for the problem formulation PP1 (20), ii) stochastic approaches (i.e. Genetic Algorithm (GA) [39]), and iii) heuristic (i.e., a two-step method that combines B&B and LP relaxation provided by Matlab optimization toolbox [40]). Then, we compare the computing time and the convergence

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

iterations of JDQT and AJDQT and we explore the QoS and the computing time under controlled solution degradation. Finally, we compare the QoS and the energy consumption of applying JDQT to solve the QoS-aware task mapping problem (QoS-OPT, i.e., PP1 (20)), and the following energy-aware task mapping problem (NRG-OPT):

$$\min_{\substack{\boldsymbol{c},\boldsymbol{q},\boldsymbol{o},\\ \boldsymbol{t}_s,\boldsymbol{h}}} \sum_{i=1}^{N}\sum_{l=1}^{L}\left(c_{il}w_{il}+\frac{h_{il}}{f_l}\right)(P_d^l+P_s^l-P_s^0)$$

s.t. (2), (3), (11), (12), (13), (14), (15), (17), (18), (19).

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.

A. Comparison with Existing Algorithms

The QoS achieved by JDQT for all tuned parameters (i.e., processor number M, task number N, and energy efficiency factor η) is shown in Fig. 5. With M, N and η parameters varying, $3 \times 9 \times 3 = 81$ instances are generated. We observe that the QoS i) increases with η under given M and N, and ii) increases with N under given M and η (as the energy supply E_s increases with N and η). However, when we fix N and η and change the value of M, the differences between the achieved QoS are small, compared with changing N and η. Although with M increasing, more tasks are allowed to be executed on different processors simultaneously. As the energy supply E_s and the task deadline, which is determined by the EOD matrix $\boldsymbol{P}$, are fixed, the QoS improvement achieved by increasing the value of M is limited. Moreover, the QoS achieved by JDQT, B&B and B&C are the same for all tuned M, N and η parameters, and, thus, JDQT also finds an optimal solution.

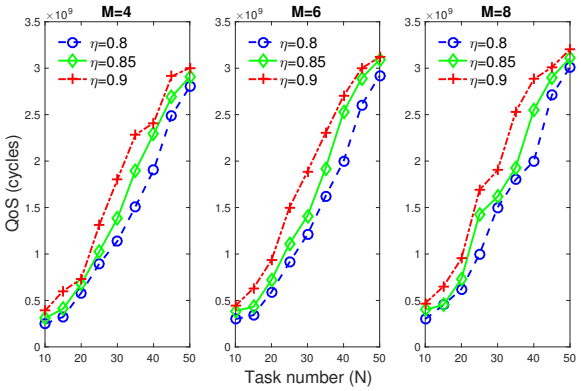

Fig. 5. QoS of JDQT with M, N and η varying.

We further compare the QoS gain of JDQT with GA and heuristic. The statistical property of QoS gain is shown in Fig. 6. The QoS gain between JDQT and GA (JDQT and heuristic) is given by $\frac{Q_J(M,N,\eta)-Q_G(M,N,\eta)}{Q_J(M,N,\eta)}$ ($\frac{Q_J(M,N,\eta)-Q_H(M,N,\eta)}{Q_J(M,N,\eta)}$), where $Q_J(M,N,\eta)$, $Q_G(M,N,\eta)$ and $Q_H(M,N,\eta)$ are the QoS achieved by JDQT, GA and heuristic under given M, N and η parameters, respectively. The box plot of "JDQT vs GA" with $\eta = 0.8$ shows the statistical property of data set $\{\frac{Q_J(M,N,0.8)-Q_G(M,N,0.8)}{Q_J(M,N,0.8)}\}$ for all tuned 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. Fig. 6 shows that i) JDQT achieves higher QoS (6.8% and 17.9% in average, respectively) than GA and heuristic, and ii) the gap between the upper and lower bounds of "JDQT vs GA" is smaller than the gap between the upper and lower bounds of "JDQT vs heuristic", which implies that the quality of the solution achieved by GA is more "stable" than the heuristic. This is because i) the aim of heuristic is to find feasible solution, and the searching process stops when an arbitrary feasible solution is found, and ii) the quality of the solution of GA can be improved through the iterations.

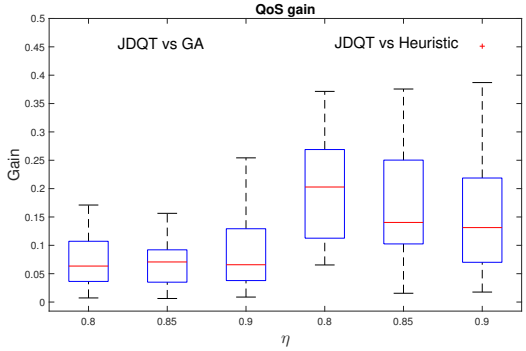

Fig. 6. QoS gain of JDQT, GA and heuristic.

Fig. 7 compares the computing time of JDQT, B&B, B&C, GA and heuristic. Note that energy efficiency factor η does not change the problem size (i.e., the number of variables and constraints), and, thus, its influence on computing time is limited. We set $\eta = 0.8$ and tune M and N parameters. Fig. 7 shows the statistical property of computing time gain between i) JDQT and B&B, ii) JDQT and B&C, iii) JDQT and GA, and iv) JDQT and heuristic for all tuned M and N parameters. Denote $T_J(M,N,\eta)$, $T_B(M,N,\eta)$, $T_C(M,N,\eta)$, $T_G(M,N,\eta)$ and $T_H(M,N,\eta)$ as the computing time of JDQT, B&B, B&C, GA and heuristic under given M, N and η parameters, respectively. The box plot of "B&B vs JDQT" with $\eta = 0.8$ shows the statistical property of data set $\{\frac{T_B(M,N,0.8)-T_J(M,N,0.8)}{T_B(M,N,0.8)}\}$ for all tuned M and N parameters. The simulation results show that when M and N increase, the computing time of JDQT, B&B, B&C and GA grows, since more variables and constraints are involved in the problem (i.e. the problem size is enlarged). For the heuristic, since it is based on a relaxed B&B method, the number of nodes still increases with problem size. Thus, the change of M and N parameters also influences the computing time of heuristic. However, the computing time increase of heuristic is much smaller than other algorithms.

As shown in Fig. 7, JDQT takes a shorter computing time than B&B (27.8% in average), B&C (19.6% in average) and GA (73.2% in average). The heuristic has a shorter computing time (81.2% in average) than JDQT. However, the quality of the solution is hard to control, as shown in Fig. 6. Although GA can solve mixed non-linear programming problem such as MINLP, the optimality of the solution is hard to guarantee. Moreover, compared with JDQT, the architecture of GA is more complex since at each iteration GA needs to generate new populations through several procedures, such as selection, reproduction, mutation and crossover. Therefore, the problem transformation from MINLP-based PP (10) to MILP-based PP1 (20) is necessary, since it can simplify the structure of the problem, and, thus, the optimal solution is much easier to find. Although B&B can optimally solve MILP problems for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

large problem sizes, B&B explores a large number of nodes to find the optimal solution. B&C, which combines the benefits of B&B and Gomory cutting scheme, can better explore optimality, efficiency and stability trade-off. Usually, B&C has a faster convergence speed than B&B. For an optimization problem, its computational complexity usually increases significantly with the number of variables and constraints. Hence, solving the smaller problems with less variables and constraints (i.e., MP and SP) iteratively is more efficient than solving a single large problem. This result is in line with the comparison of [38] where the decomposition-based method is faster than B&B and B&C for solving lager problem instances.

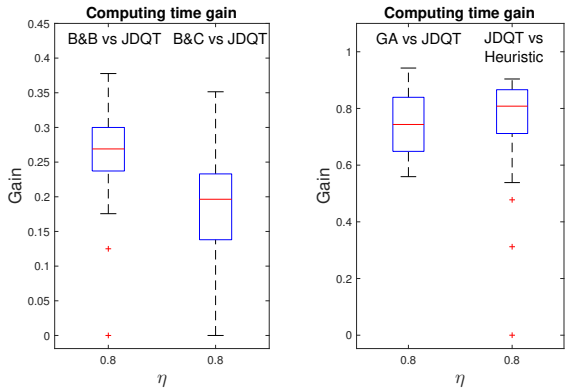

Fig. 7. Computing time gain of JDQT, B&B, B&C, GA and heuristic.

B. Comparison between JDQT and AJDQT

In this section, we first explore the convergence of the proposed two-layer subgradient-based algorithm used in AJDQT to solve the MP1 (38) and the SP (29) with respect to the outer-layer iterations, which are the most dominant ones as they determining the convergence criteria, as shown in the proof of *Theorem 3.3*. Fig. 8 shows the convergence of the objective function to the optimal solution with respect to the number of the outer-layer iterations for the MP1, with $\zeta = 0.01$, $\sigma = 0.1$, $\eta = 0.8$, $M = 4$ and $N = 20$. The results are similar then for the SP. From the experimental results, the optimal solution is found usually within 15 outer-layer iterations.

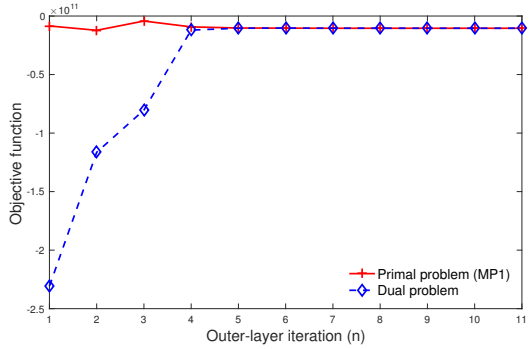

Fig. 8. Convergence of two-layer subgradient-based algorithm.

The simulation results show that the solutions of PP1 (20) achieved by JDQT and AJDQT are the same, for stopping criteria $\epsilon = 0.01$ and all tuned M, N and η parameters. Therefore, AJDQT is also able to find an optimal solution. The computing time of JDQT and AJDQT is compared in Fig. 9. Since energy efficiency factor η does not change problem size, we set $\eta = 0.8$ and tune M and N parameters. From it, we can see that i) when increasing M and N, the computing time of JDQT and AJDQT grows, and ii) under the same M and N parameters, AJDQT has a shorter computing time than JDQT (17.6% in average). This is because at each iteration the MP of AJDQT is relaxed to an LP, but the MP of JDQT is still an ILP. LP is much easier to solve than ILP, especially when the problem size is large. We also observe that the computational complexity of AJDQT is largely influenced by the algorithm used to find the feasible solution to the MP (i.e., FP algorithm in our case). When FP cannot find a feasible solution $\overline{\boldsymbol{x}}(m)$ for the first time, we need to repeat FP again until an arbitrary feasible solution is found.

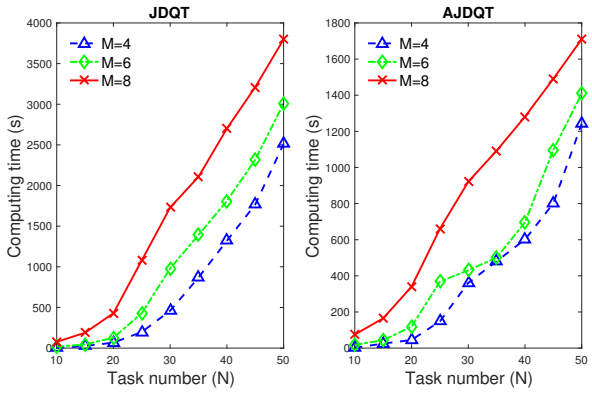

Fig. 9. Computing time of JDQT and AJDQT with M and N varying.

The convergence iterations of JDQT and AJDQT are compared in Fig. 10, where the convergence iteration is defined as the number of iterations to achieve $|\Phi_u(m) - \Phi_l(m)| \leq \epsilon$. We set $\epsilon = 0.01$, $\eta = 0.8$ and tune M and N parameters. From Fig. 10, we can see that i) the convergence iterations of JDQT and AJDQT increase with M and N parameters. Since more variables and constraints are added into the problem, the feasible region of the problem becomes more complex, and, thus, a higher number of iterations is required to search the optimal solution; ii) under the same M and N parameters, the difference of convergence iterations between JDQT and AJDQT is small. Usually, the difference is within several iterations, and this difference comes from twofold: i) the initial solution $\boldsymbol{x}(0)$ of the MP is generated randomly (no matter for JDQT or AJDQT), as long as it satisfies the constraints (2) and (3) (the value of initial solution $\boldsymbol{x}(0)$ only influences convergence iteration but not the convergence of the algorithm), and ii) the feasible solution of the MP (i.e., $\overline{\boldsymbol{x}}(m)$), rather than the optimal solution of the MP, is inserted into the iteration between the MP and the SP of AJDQT, and, thus, the lower bound sequence of AJDQT is not strictly increasing, compared with the lower bound sequence of JDQT. Note that i) the upper bound sequence of AJDQT is still decreasing (as the upper bound update processes of JDQT and AJDQT are the same), and ii) the feasibility or infeasibility constraints of AJDQT exclude the non-optimal and the infeasible solutions rather than the optimal solution. The optimality of the solution achieved by AJDQT is guaranteed (see *Theorem 3.2*). The characteristics of the lower and the upper bounds of JDQT and AJDQT imply that the convergence iteration of AJDQT may be larger than JDQT, but the computing time of AJDQT at each iteration is smaller than JDQT, since AJDQT has a simpler structure.

In Fig. 11, we explore the behavior (i.e., computing time, convergence iteration and QoS) achieved by JDQT and AJDQT with stopping criteria ϵ varying, where we set $M = 4$, $N = 20$ and $\eta = 0.8$. To better evaluate the influence of ϵ exerts on algorithm performance, the gap between the lower and the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2018.2852005, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

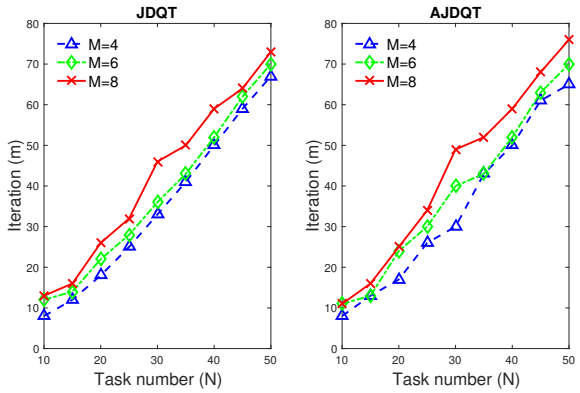

Fig. 10. Convergence iterations of JDQT and AJDQT with M and N varying.

upper bounds at the m^{th} iteration is calculated by $\frac{\Phi_u(m)}{\Phi_l(m)}$, and ϵ is tuned from 1 to 2.5 with a step of 0.5. From Fig. 11, we can see that the computing time, convergence iteration and QoS decrease when ϵ increases. Note that except $\epsilon = 1$, the solutions found by JDQT and AJDQT are non-optimal. In the non-optimal cases, the QoS achieved by JDQT is not always larger than AJDQT under the same ϵ, but it is always within the upper and lower bounds. If we just want to find a feasible solution, we can run JDQT (AJDQT) and stop the iteration when the fist time that the DSP has a bounded solution (assume that at the m^{th} iteration). This means the optimal task scheduling and adjustment decisions $\boldsymbol{y}(m)$ are found under the given task-to-processor allocation and frequency-to-task assignment decisions $\boldsymbol{x}(m)$. Since $\boldsymbol{x}(m)$ is not the optimal solution, $(\boldsymbol{x}(m), \boldsymbol{y}(m))$ is a feasible solution of the PP1 (20). Based on the lower bound $\Phi_l(m)$ and the upper bound $\Phi_u(m)$ under given solution $(\boldsymbol{x}(m), \boldsymbol{y}(m))$, we can say that this solution is $|\Phi_u(m) - \Phi_l(m)|$-optimal. Similarly, if an ϵ is given in advance, we can find the corresponding solution by iteratively solving the MP and SP of JDQT (AJDQT) until i) the gap between the upper and the lower bounds of JDQT (AJDQT) is smaller than ϵ, and ii) the DSP is feasible under the temporary solution given by the MP.

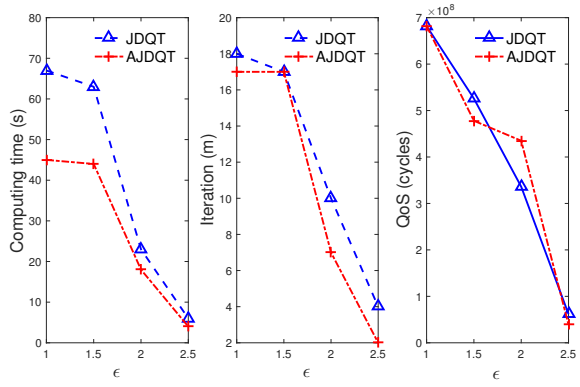

Fig. 11. Quality of solution achieved by JDQT and AJDQT with ϵ varying.

C. QoS-Aware vs Energy-Aware Task Mapping

Fig. 12 compares the QoS gain and the energy consumption gain of QoS-OPT with NRG-OPT for all tuned M, N and η parameters. Denote $Q_E(M,N,\eta)$, $E_J(M,N,\eta)$ and $E_E(M,N,\eta)$ as the QoS achieved by NRG-OPT, and the consumed energy of QoS-OPT and NRG-OPT under given M, N and η parameters, respectively. The box plot of "QoS-OPT vs NRG-OPT" with title "QoS gain" ("Energy consumption gain") shows the statistical property of data set $\{\frac{Q_J(M,N,\eta)-Q_E(M,N,\eta)}{Q_J(M,N,\eta)}\}$ ($\{\frac{E_J(M,N,\eta)-E_E(M,N,\eta)}{E_J(M,N,\eta)}\}$) for all tuned M, N and η parameters. From Fig. 12, we can see that when using NRG-OPT to perform IC-task mapping, the QoS is always equal to 0, which is obviously lower than the QoS achieved by QoS-OPT. This is normal since, if the mandatory subtask of a task is fixed and given in advance, the smaller the optional subtask, the lower the energy consumed to execute this task. On the other hand, Fig. 12 also shows that QoS-OPT consumes more energy than NRG-OPT (60.3% in average), since QoS-OPT maximizes QoS and therefore executes more optional subtasks than NRG-OPT. However, the consumed energy of QoS-OPT is always no more than the supplied energy E_s, as the energy supply constraint (9) must be satisfied. Hence, using QoS-OPT to perform IC-task mapping can provide a better balance between QoS-enhancing and energy-utilizing. In fact, when the system is in the low energy state, QoS-OPT is equal to NRG-OPT since only mandatory subtasks can be used to perform task mapping.

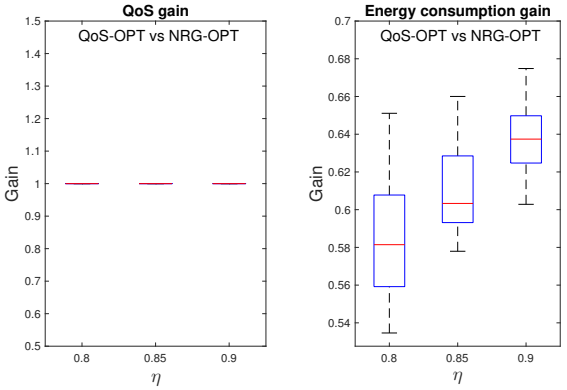

Fig. 12. QoS and energy consumption gains of QoS-OPT and NRG-OPT.

V. Conclusion

In this paper, we address the problem of IC-tasks mapping on DVFS-enabled homogeneous multicore platforms, with the goal of maximizing system QoS without violating the real-time and energy constraints. We first develop an MINLP model to describe this joint-design problem. And then, we propose an MILP description of this model without performance degradation. By doing so, we avoid high computational complexity and we use a simpler model to find an optimal solution. To optimally solve the MILP problem, we propose a JDQT algorithm. It reduces the computational complexity by iterating two smaller, but highly correlated, subproblems: an MP problem for task-to-processor allocation and frequency-to-task assignment, and a SP problem for task scheduling and task adjustment. We prove that the proposed JDQT algorithm converges to the optimal solution through finite number of iterations. To further reduce the computational complexity of JDQT algorithm, we propose an AJDQT algorithm. We prove that by relaxing the MP problem of JDQT algorithm to an LP problem, the computing time can be reduced but the optimality of solution is still guaranteed. The results show that the desired system performance can be achieved by the proposed JDQT and AJDQT algorithms. Moreover, we have shown that the stopping criteria of JDQT and AJDQT algorithms can be used as a control parameter to trade-off QoS and algorithm computing time.

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] 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.

[2] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: survey of current and emerging trends," in *Proc. ACM DAC*, 2013, pp. 1–10.

[3] I. Mendez-Diaz, J. Orozco, R. Santos, and P. Zabala, "Energy-aware scheduling mandatory/optional tasks in multicore real-time systems," *Int. Trans. Oper. Res.*, vol. 24, no. 12, pp. 173–198, 2017.

[4] 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.

[5] 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.

[6] Y. Zhang, Y. Wang, and H. Wang, "Energy-efficient task scheduling for DVFS-enabled heterogeneous computing systems using a linear programming approach," in *Proc. IEEE IPCCC*, 2016, pp. 1–8.

[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] A. Emeretlis, G. Theodoridis, P. Alefragis, and N. Voros, "A logic-based Benders decomposition approach for mapping applications on heterogeneous multicore platforms," *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 1, pp. 19:1–19:28, 2016.

[9] A. Davare, J. Chong, Q. Zhu, D. M. Densmore, and A. L. Sangiovanni-Vincentelli, "Classification, customization, and characterization: using MILP for task allocation and scheduling," University of California, Berkeley, Tech. Rep., 2006.

[10] 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.

[11] S. A. Ishak, H. Wu, and U. U. Tariq, "Energy-aware task scheduling on heterogeneous NoC-based MPSoCs," in *Proc. IEEE ICCD*, 2017, pp. 165–168.

[12] S. Tosun, "Energy- and reliability-aware task scheduling onto heterogeneous MPSoC architectures," *J. Supercomput.*, vol. 62, no. 1, pp. 265–289, 2012.

[13] P. Zhou and W. Zheng, "An efficient biobjective heuristic for scheduling workflows on heterogeneous DVS-enabled processors," *J. Appl. Math.*, vol. 2014, pp. 1–15, 2014.

[14] 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.

[15] C. Rusu, R. Melhem, and D. Mossé, "Maximizing rewards for real-time applications with energy constraints," *ACM Trans. Embed. Comput. Syst.*, vol. 2, no. 4, pp. 537–559, 2003.

[16] 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.

[17] 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.

[18] T. Wei, J. Zhou, K. Cao, P. Cong, M. Chen, G. Zhang, X. S. Hu, and J. Yan, "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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: a survey," *Surveys in Oper. Res. Manag. Sci.*, vol. 17, no. 2, pp. 97–106, 2012.

[25] T. Chantem, X. S. Hu, and R. P. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 10, pp. 1884–1897, 2011.

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

[27] L. P. Qian, Y. J. A. Zhang, Y. Wu, and J. Chen, "Joint base station association and power control via Benders' decomposition," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1651–1665, 2013.

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

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

[30] C. H. Papadimitriou, "On the complexity of integer programming," *J. ACM*, vol. 28, no. 4, pp. 765–768, 1981.

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

[32] M. Fischetti, F. Glover, and A. Lodi, "The feasibility pump," *Math. Program.*, vol. 104, no. 1, pp. 91–104, 2005.

[33] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Cross-layer congestion control, routing and scheduling design in Ad Hoc wireless networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–13.

[34] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Trans. Signal Process.*, vol. 57, no. 2, pp. 451–462, 2009.

[35] M. Ito and M. Fukuda, "A family of subgradient-based methods for convex optimization problems in a unifying framework," *Optim. Method and Soft.*, vol. 31, no. 5, pp. 952–982, 2016.

[36] S. Boyd and J. Mattingley, "Branch and bound methods," *Notes for EE364b, Stanford University*, pp. 1–11, 2007.

[37] K. Pang, V. Fresse, and S. Yao, "Communication-aware branch and bound with cluster-based latency-constraint mapping technique on network-on-chip," *J. Supercomput.*, vol. 72, no. 6, pp. 2283–2309, 2016.

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

[39] 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.

[40] https://mathworks.com/help/optim/ug/intlinprog.html?s_tid=gn_loc_drop.

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 served as a Guest Editor for IEEE Access and Journal of Computer Networks and Communications. He also served as a TPC Member for WF-5G'18, GLOBECOM-MWN'18 and COMNETSAT'18.

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.