Approximation-aware Task Deployment on Asymmetric Multicore Processors

Lei Mo, Angeliki Kritikakou, Olivier Sentieys

Approximation-aware Task Deployment on Asymmetric Multicore Processors

Lei Mo*, Angeliki Kritikakou*, and Olivier Sentieys*
*Univ Rennes, INRIA, CNRS, IRISA, France
Email: lei.mo@inria.fr, {angeliki.kritikakou, olivier.sentieys}@irisa.fr

***Abstract*—Asymmetric Multicore Processors (AMP) are a very promising architecture to deal efficiently with the wide diversity of applications. In real-time application domains, in-time approximated results are preferred than accurate – but too late – results. In this work, we propose a deployment approach that exploits the heterogeneity provided by AMP architectures and the approximation tolerance provided by the applications, so as to increase as much as possible the quality of the results under given energy and timing constraints. Initially, an optimal approach is proposed based on problem linearization and decomposition. Then, a heuristic approach is developed based on iteration relaxation of the optimal version. The obtained results show 16.3% reduction in the computation time for the optimal approach compared to the conventional optimal approaches. The proposed heuristic approach is about 100 times faster at the cost of a 29.8% QoS degradation in comparison with the optimal solution.**

I. Introduction

Due to the wide range of nowaday's application domains, such as multimedia, encryption and network, the applications require different system resources, impose different design constraints and have different optimization objectives. Conventional Symmetric Multi-core Processors (SMP) have become a less promising option, as they cannot simultaneously meet these – often conflicting – requirements and goals. An overprovisioned core wastes area and energy, whereas an underprovisioned core fails to provide the required performance [1]. Thus, Asymmetric Multi-core Processors (AMP) are proposed to deal with the application diversity. AMPs provide the required heterogeneity, since they consist of cores that differ in microarchitectural features, such as pipeline design, issue/fetch width, and cache hierarchy – and not merely in frequency/voltage, as SMPs [1].

In real-time application domains, approximated results obtained in time are preferred than accurate results obtained after the deadline. For instance, in a real-time video application, at each period, an imperfect, but acceptable, quality image is initially produced from the received data. Then, this image can be further refined depending on the available system resources [2]. Similar applications can be found in many other areas like mobile target tracking, real-time heuristic research and control engineering [3], [4]. In order to maximize the quality of the obtained application results and at the same time meet the design constraints, proper deployment approaches have to explore both the architecture heterogeneity and the approximation tolerance of the applications.

The majority of the deployment approaches on multicore processors (SMP/AMP) focuses on precise computation tasks [5]–[8]. Usually these approaches aim to minimize the energy consumption under real-time constraints. On the other hand, the deployment approaches that focus on tasks that tolerate approximation, such as Imprecise Computation (IC) tasks [9], aim to maximize the Quality of Service (QoS) under energy and real-time constraints. The state-of-the-art on IC task deployment approaches can be classified based on whether: 1) the tasks are independent or dependent, 2) the platform is an SMP or an AMP, and 3) the solution is optimal or heuristic. Table I positions our work with respect to several representative papers from the literature. Most existing works deal with the independent IC tasks deployment problems on AMP/SMP. Works on deploying dependent IC tasks on AMP/SMP are rare and they solve different problems compared to our work, e.g., task-to-processor allocation is fixed and given upfront [3], [10], and no task migration is considered in [11]. The extension of the optimal deployment methods from dependent to independent tasks and from SMPs to AMPs is not straightforward, as additional nonlinear constraints are usually introduced into the problem. On the other hand, the heuristics of previous works usually adopt multi-step optimizations, e.g., [4], [12], [13] and they require that the variables can be solved separately. Hence, they need to be re-designed when the problem formulation changes. In contrast, the heuristic proposed in this paper can solve general Mixed-Integer Linear Programming (MILP) problem.

TABLE I
Classification of some IC task deployment approaches

Reference	*Task*				*Platform*		*Solution*	
	Dep.	Indep.	Alloc.	Migr.	AMP	SMP	Opt.	Heur.
Proposed	√		√	√	√		√	√
[2]		√				√	√	
[3]	√				√			√
[4]		√	√			√		√
[10]	√					√		√
[11]	√		√			√	√	
[12]		√	√		√			√
[13]		√	√			√	√	√
[14]		√	√		√			√

The goal of this work is to address the problem of dependent IC tasks deployment on AMP, so as to increase the QoS of the obtained results and at the same time meet the timing and energy constraints. Our first contribution is the formulation of the deployment problem as a Mixed-Integer Non-Linear Programming (MINLP) and its safe transformation into an MILP. Our second contribution is an Optimal Deployment Approach (ODA) based on Benders decomposition [15] and the algorithm's convergence analysis. Our third contribution is a Heuristic Deployment Approach (HDA), which relaxes the number of required iterations between the ODA subproblems. Finally, the extended experimental results show 16.3% less computation time for the ODA compared to the conventional state-of-the-art optimal approaches. HDA runs ∼100 times

faster than ODA with a cost of 29.8% in QoS.

The rest of the paper is organized as follows. Section II introduces the system model. Section III presents the problem formulation. Section IV provides the problem linearization method. Section V and Section VI propose the ODA and HDA algorithms, respectively. Section VII evaluates the performance of the proposed methods. Section VIII concludes this study.

II. System Model

1) Task Model:

Definition 2.1: (*IC task [9]*) A task can be logically decomposed into a mandatory subtask and an optional subtask. The mandatory subtask must be completed before the deadline to produce the acceptable result, whereas the optional subtask can be left incomplete at the cost of reduced quality.

We consider a task set $\mathcal{T}$ consisting of N dependent and periodic IC tasks $\{\tau_1, \ldots, \tau_N\}$ released at time 0 and sharing a common hyper-period H. The task set $\mathcal{T}$ is modeled by a Directed Acyclic Graph (DAG) $G(V, E)$, where the nodes V correspond to the IC tasks and the edges E indicate the data dependencies between the tasks. Each task τ_l is described by a tuple $\{o_l, M_l, O_l, t_l^s, d_l, g_l\}$. A task τ_l is decomposed into a mandatory subtask and an optional subtask characterized in number of execution cycles, i.e. M_l and o_l, respectively. O_l is the maximum optional cycles of task τ_l. M_l and O_l are measured in Worst Case Execution Cycles (WCECs). t_l^s, d_l and g_l are the start time, the deadline and the period of task τ_l, respectively. We assume that the tasks are non-preemptive and H is the least common multiple of $\{g_1, \ldots, g_N\}$.

2) Platform Model: We consider an AMP platform with M processors $\{\theta_1, \ldots, \theta_M\}$. Each processor θ_k is characterized by a given supply voltage and frequency pair (v_k, f_k). According to the characteristics of many AMP platforms (e.g., ARM big.LITTLE) [6], we introduce the concept of *cluster*. M processors are divided into R clusters $\{\mathcal{W}_1, \ldots, \mathcal{W}_R\}$. Each cluster $\mathcal{W}_r$ consists of a set of symmetric processors that have the same frequency characteristics (e.g., minimal, maximal and operating frequencies). Therefore, $v_k = v_{k'}$ and $f_k = f_{k'}$, if $k, k' \in \mathcal{W}_r$. In addition, we define $\lambda_{l,k} \in (0, 1]$ as the execution efficiency of processor θ_k, when it executes task τ_l [5]. Correspondingly, the Worst Case Execution Time (WCET) of task τ_l, when it is executed on processor θ_k, is calculated as $\frac{M_l + o_l}{f_k \lambda_{l,k}}$.

3) Energy Model: Processors can operate in two modes: *idle* and *active*. A processor is said to be in the active mode, if the processor currently executes a task. Otherwise, it is in the idle mode. The power consumption of a processor θ_k is expressed as $P_k^c = P_k^s + P_k^d$, where $P_k^s = C_k^s v_k^{\rho_k}$ is the static power of the processor ready to execute (being either on the active or idle mode), $P_k^d = C_k^d f_k v_k^2$ is the dynamic power of task execution, and C_k^s, ρ_k and C_k^d are constants depending on processor type [6]. We assume that when a processor has no task to execute, it goes into idle mode. The transition time and energy overhead is considered very small compared to the one required to execute a task and is assumed to be incorporated into the execution time and energy of the task [5]. This power consumption model is widely adopted by existing works [5], [7], [12]. The system is *energy-constrained* [12] in the sense that: 1) the energy budget E_s is fixed and cannot be replenished during the hyper-period H, and 2) with E_s budget, all mandatory subtasks $\{M_1, \ldots, M_N\}$ are ensured to finish, but not all optional subtasks $\{O_1, \ldots, O_N\}$ can complete their executions.

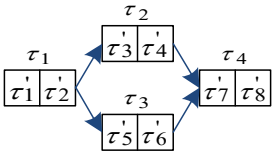


Fig. 1. Task graph of illustration example.

III. Problem Formulation

Let $\mathcal{N} \triangleq \{1, \ldots, N\}$, $\mathcal{N}' \triangleq \{1, \ldots, 2N\}$, $\mathcal{M} \triangleq \{1, \ldots, M\}$ and $\mathcal{R} \triangleq \{1, \ldots, R\}$. Denote τ'_{2l-1} and τ'_{2l} as the mandatory and the optional subtasks of task τ_l ($\forall l \in \mathcal{N}$), respectively. Based on the task and the platform models mentioned above, we consider 1) task level migration [6] on the AMP (i.e., task τ_l in one cluster can migrate to the other cluster after the execution of the mandatory subtask, and 2) subtask τ'_{2l} starts running only after subtask τ'_{2l-1} is finished. Note that the subtask set $\{\tau'_{2l-1}, \tau'_{2l}\}$ ($\forall l \in \mathcal{N}$) can be rewritten as $\{\tau'_i\}$ ($\forall i \in \mathcal{N}'$). The dependency between the subtasks $\{\tau'_i\}$ in one hyper-period H are described by a symmetric Subtask Dependency Decision (SDD) matrix $\boldsymbol{s} \triangleq [s_{i,j}]$. If $s_{i,j} = 1$, subtasks τ'_i and τ'_j are dependent with each other, otherwise, $s_{i,j} = 0$. In addition, we introduce an Execution Order Decision (EOD) matrix $\boldsymbol{p} \triangleq [p_{i,j}]$. If $p_{i,j} = 1$, subtask τ'_i precedes subtask τ'_j and τ'_j is the closest task of τ'_i, otherwise, $p_{i,j} = 0$. Fig. 1 shows the DAG of an illustration example with 4 dependent IC tasks. The corresponding SDD matrix $\boldsymbol{s}$ and EOD matrix $\boldsymbol{p}$ are as follows:

$$\boldsymbol{s} = \begin{bmatrix} 0&1&0&0&0&0&0&0 \\ 1&0&1&0&1&0&0&0 \\ 0&1&0&1&0&0&0&0 \\ 0&0&1&0&0&0&1&0 \\ 0&1&0&0&0&1&0&0 \\ 0&0&0&0&1&0&1&0 \\ 0&0&0&1&0&1&0&1 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}, \boldsymbol{p} = \begin{bmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}.$$

The problem consists of an objective function to maximize the system QoS subject to a set of real-time and energy constraints. We have to simultaneously cope with the constraints of 1) task level mitigation, 2) subtask dependency, 3) task deadline, and 4) energy supply. Under these constraints, we determine 1) on which processor should the subtask be executed (*subtask allocation and mitigation*), and 2) when should the subtask start and end (*subtask scheduling and optional subtask adjustment*). To formulate the task deployment problem, we introduce the following variables: 1) define $\boldsymbol{q} \triangleq [q_{i,k}]$ and let the binary variable $q_{i,k} = 1$, if subtask τ'_i is assigned to processor θ_k, otherwise, $q_{i,k} = 0$; 2) define $\boldsymbol{u} \triangleq [u_{i,j}]$ and let the binary variable $u_{i,j} = 1$, if subtask τ'_i precedes subtask τ'_j (i.e., τ'_i starts after the end time of τ'_i), otherwise, $u_{i,j} = 0$; 3) define $\boldsymbol{t}^s \triangleq [t_i^s]$ and let the continuous variable t_i^s be the start time of subtask τ'_i; 4) define $\boldsymbol{o} \triangleq [o_l]$ and let the continuous variable o_l be the optional subtask of task τ_l. It has been shown in [2], [3] that in several application domains (e.g., image processing), a linear function models the cases where the system QoS increases uniformly during the optional execution. We introduce a linear function $g_l(o_l)$ for task τ_l, and our aim is to maximize QoS function $\sum_{l \in \mathcal{N}} g_l(o_l)$. The Primal Problem (PP) is given by

$$\textbf{PP}: \min_{\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{o}, \boldsymbol{t}^s} -\sum\nolimits_{l \in \mathcal{N}} g_l(o_l), \tag{1}$$

$$\text{s.t. } \boldsymbol{C}_1 - \boldsymbol{C}_7.$$

The constraint descriptions are as follows:

- Subtask allocation constraint $\boldsymbol{C}_1 : \sum_{k\in\mathcal{M}} q_{i,k} = 1$ ($\forall i \in \mathcal{N}'$). Each subtask τ'_i is executed on only one processor.
- Task migration constraint $\boldsymbol{C}_2 : q_{2l-1,k} + q_{2l,k'} \leq 1$ ($\forall l \in \mathcal{N}$, $\forall k, k' \in \mathcal{W}_r$, $k \neq k'$, $\forall r \in \mathcal{R}$). The migration of task τ_l is performed among different clusters, as the processors in the same cluster are homogeneous. Similar to the previously published work [6], we consider task migration occurs at no cost or penalty.
- Subtask non-preemptive constraints $\boldsymbol{C}_3 : t^e_i \leq t^s_j + (2 - q_{i,k} - q_{j,k})H + (1 - u_{i,j})H$ ($\forall i, j \in \mathcal{N}'$, $s_{ij} = 0$, $i \neq j$, $\forall k \in \mathcal{M}$) and $\boldsymbol{C}_4 : t^e_j \leq t^s_i + (2 - q_{i,k} - q_{j,k})H + u_{i,j}H$ ($\forall i, j \in \mathcal{N}'$, $s_{ij} = 0$, $i \neq j$, $\forall k \in \mathcal{M}$). In $\boldsymbol{C}_3$ and $\boldsymbol{C}_4$, $t^e_{2l-1} = t^s_{2l-1} + \sum_{k\in\mathcal{M}} \frac{q_{2l-1,k}M_l}{f_k\lambda_{2l-1,k}}$ and $t^e_{2l} = t^s_{2l} + \sum_{k\in\mathcal{M}} \frac{q_{2l,k}o_l}{f_k\lambda_{2l,k}}$ ($\forall l \in \mathcal{N}$) are the end time of subtasks τ'_{2l-1} and τ'_{2l}, respectively. If subtasks τ'_i and τ'_j are executed on the same processor θ_k (i.e., $q_{i,k} = q_{j,k} = 1$), $\boldsymbol{C}_3$ and $\boldsymbol{C}_4$ are meaningful, else, $\boldsymbol{C}_3$ and $\boldsymbol{C}_4$ are always satisfied. With $q_{i,k} = q_{j,k} = 1$, if subtask τ'_i precedes subtask τ'_j (i.e., $u_{i,j} = 1$), we have $t^e_i \leq t^s_j$ and $t^e_j \leq t^s_i + H$; if $u_{i,j} = 0$, we have $t^e_i \leq t^s_j + H$ and $t^e_j \leq t^s_i$.
- Subtask dependency constraint $\boldsymbol{C}_5 : t^s_j + (1 - p_{i,j})H \geq t^s_i + p_{i,j}(t^e_i - t^s_i)$ ($\forall i, j \in \mathcal{N}'$, $i \neq j$). If $p_{ij} = 1$, subtask τ'_i precedes subtask τ'_j and τ'_j is the closest task of τ'_i. $\boldsymbol{C}_5$ is meaningful for task start time t^s_i and t^s_j (i.e., $t^s_j \geq t^e_i$). If $p_{ij} = 0$, $\boldsymbol{C}_5$ can be ignored since the inequality $t^s_j + H \geq t^s_i$ is always true.
- Task deadline constraint $\boldsymbol{C}_6 : t^e_{2l} = t^s_{2l} + \sum_{k\in\mathcal{M}} \frac{q_{2l,k}o_l}{f_k\lambda_{2l,k}} \leq d_l$ ($\forall l \in \mathcal{N}$). Each task τ_l should be finished before its predefined deadline d_l ($0 \leq d_l \leq H$).
- Energy constraint $\boldsymbol{C}_7 : \sum_{k\in\mathcal{M}}[t^d_k P^c_k + (H - t^d_k)P^s_k] \leq E_s$. The total energy consumed by M processors during the hyper-period H should not exceed the energy supply E_s. Let $t^d_k = \sum_{l\in\mathcal{N}} \left(\frac{q_{2l-1,k}M_l}{f_k\lambda_{2l-1,k}} + \frac{q_{2l,k}o_l}{f_k\lambda_{2l,k}}\right)$ be the time required to execute all the tasks assigned to processor θ_k. Therefore, $H - t^d_k$ is the time of processor θ_k being in the idle state during the hyper-period H.

For tractability reasons, we consider o_i as continuous variables. When PP is solved, we round the result down. This impact of one cycle is negligible, since the tasks usually execute typically hundreds of thousands of cycles [16].

IV. MINLP Linearization

Due to the product $q_{2l,k}o_l$ of the optimization variables in the constraints $\boldsymbol{C}_2 - \boldsymbol{C}_7$, PP is an MINLP problem, which is $\mathcal{NP}$-hard. Due to page limit, the proofs of the lemmas are omitted.

Lemma 4.1: Given constants $s_1, s_2 > 0$ and two constraint spaces $S_1 = \{[h, b, x] | h = bx, -s_1 \leq x \leq s_2, b \in \{0, 1\}\}$ and $S_2 = \{[h, b, x] | -bs_1 \leq h \leq bs_2, h + bs_1 - x - s_1 \leq 0, h - bs_2 - x + s_2 \geq 0, b \in \{0, 1\}\}$, then $S_1 \leftrightarrows S_2$.

Since $q_{2l,k} \in \{0, 1\}$ and $0 \leq o_l \leq O_l$, based on *Lemma 4.1*, an *auxiliary* variable (continuous) $h_{2l,k}$ is introduced into PP to replace the nonlinear item $q_{2l,k}o_l$. Thus, PP is rewritten as

$$\begin{aligned} \textbf{PP1}: \min_{\boldsymbol{q},\boldsymbol{u},\boldsymbol{o},\boldsymbol{h},\boldsymbol{t}^s} & -\sum\nolimits_{l\in\mathcal{N}} g_l(o_l), \\ \text{s.t.} \quad & \boldsymbol{C}_1 - \boldsymbol{C}_{10}, \end{aligned} \tag{2}$$

where $t^e_{2l} = t^s_{2l} + \sum_{k\in\mathcal{M}} \frac{h_{2l,k}}{f_k\lambda_{2l,k}}$ and $t^d_k = \sum_{l\in\mathcal{N}} \left(\frac{q_{2l-1,k}M_l}{f_k\lambda_{2l-1,k}} + \frac{h_{2l,k}}{f_k\lambda_{2l,k}}\right)$. $\boldsymbol{C}_8 : h_{2l,k} \leq q_{2l,k}O_l$, $\boldsymbol{C}_9 : h_{2l,k} \leq o_l$ and $\boldsymbol{C}_{10} : h_{2l,k} + (1 - q_{2l,k})O_l \geq o_l$ ($\forall l \in \mathcal{N}$, $\forall k \in \mathcal{M}$) are the additional constraints introduced by the MINLP linearization. The PP1 is an MILP problem, as the binary variables $(\boldsymbol{q}, \boldsymbol{u})$ and the continuous variables $(\boldsymbol{o}, \boldsymbol{h}, \boldsymbol{t}^s)$ are coupled with each other linearly, which is much easier to solve than the PP.

Remark 4.1: Since 1) the objective functions of PP and PP1 are the same, and 2) the variable replacement does not change the feasible region of the problem ($S_1 \leftrightarrows S_2$), solving PP1 is equivalent to solving PP, i.e., the optimal values of the objective functions of PP and PP1 are the same.

V. Optimal Task Deployment Algorithm

In this section, we propose an optimal algorithm ODA to efficiently solve PP1. Using concise notions, the objective function and constraints of PP1 can be rewritten as

$$\begin{aligned} \textbf{PP2}: \min_{\boldsymbol{x},\boldsymbol{y}} \ & \Phi = \boldsymbol{f}^T\boldsymbol{y} \\ \text{s.t.} \ & \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1, \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2, \end{aligned} \tag{3}$$

where $\boldsymbol{x}$ ($\boldsymbol{y}$) is the vector of binary (continuous) variables. $\boldsymbol{f}$ is the vector of the objective function coefficient. $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are the constraint coefficient matrices. $\boldsymbol{b}_1$ ($\boldsymbol{b}_2$) is a u (v)-dimensional vector. Instead of simultaneously solving the binary variables $\boldsymbol{x}$ and the continuous variables $\boldsymbol{y}$, ODA: 1) decomposes PP2 into two smaller subproblems with less variables: an ILP-based *Master Problem* (MP) with binary variables $\boldsymbol{x}$ and an LP-based *Slave Problem* (SP) with continuous variables $\boldsymbol{y}$, and 2) solves them by using the solution of one to the other. By doing so, the computational complexity is significantly reduced.

1) MP and SP formulation: Based on the structure of PP2, at the k^{th} iteration, the MP and the SP are formulated as

$$\begin{aligned} \textbf{MP}: \Phi_l(k) = \min_{\boldsymbol{x},\hat{\Phi}} \ & \hat{\Phi} \\ \text{s.t.} \ & \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1, \boldsymbol{C}_{11}, \boldsymbol{C}_{12}, \end{aligned} \tag{4}$$

$$\begin{aligned} \textbf{SP}: \Phi_u(k) = \min_{\boldsymbol{y}\succeq 0} \ & \boldsymbol{f}^T\boldsymbol{y} \\ \text{s.t.} \ & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2, \end{aligned} \tag{5}$$

where $\boldsymbol{x}(k)$ is the optimal solution of the MP at the k^{th} iteration. $\boldsymbol{C}_{11}$ is the *Feasibility Constraints* (FCs), $\hat{\Phi} \geq \boldsymbol{\lambda}(i)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ ($\forall i = 1, \ldots, m$) and $\boldsymbol{C}_{12}$ is the *Infeasibility Constraints* (ICs), $0 \geq \hat{\boldsymbol{\lambda}}(j)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ ($\forall j = 1, \ldots, n$), where $m + n = k$.

Remark 5.1: Since the objective function of the PP2 only contains the continuous variables $\boldsymbol{y}$, and the MP only considers the binary variables $\boldsymbol{x}$, an *auxiliary* variable (continuous) $\hat{\Phi}$ is introduced into the MP as the objective function. $\hat{\Phi}$ and Φ have the same physical meaning, as shown in the *Proof* 5.1. Although the MP includes the continuous variable $\hat{\Phi}$, this problem can be solved by only considering the binary variables $\boldsymbol{x}$ [13].

Lemma 5.1: Solving the MP (SP), we obtain a lower bound $\Phi_l(k)$ (upper bound $\Phi_u(k)$) of the optimal value of PP2, Φ^*.

2) MP and SP Iterations: At the initial iteration, i.e., $k = 0$, we set $\{\boldsymbol{x}(0) | \boldsymbol{A}\boldsymbol{x}(0) \preceq \boldsymbol{b}_1\}$, $\Phi_l(0) = -\infty$ and $\Phi_u(0) = +\infty$. Denote ε as a small positive tolerance. The iteration stops when $\Phi_u(k) - \Phi_l(k) \leq \varepsilon$.

(a) Step 1: Solve the Dual Slave Problem (DSP)

$$\mathbf{DSP}: \max_{\boldsymbol{\lambda}\succeq 0} \ \boldsymbol{\lambda}^T(\boldsymbol{C}\boldsymbol{x}(k)-\boldsymbol{b}_2) \quad \text{s.t.} \quad \boldsymbol{f}+\boldsymbol{D}'\boldsymbol{\lambda}\succeq 0. \tag{6}$$

where $\boldsymbol{\lambda} \triangleq [\lambda_i]$ are the Lagrange multipliers.

(b) Step 2: Based on the solution of the DSP, a new constraint is generated. Case 1: If DSP is *infeasible*, PP2 has no feasible solution. Case 2: If DSP has a *bounded solution* $\boldsymbol{\lambda}(k)$, a new FC: $\hat{\Phi} \geq \boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)$ is generated. Case 3: If DSP has an *unbounded solution*, i.e., $\boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x}(k)-\boldsymbol{b}_2)=+\infty$, a new IC: $0 \geq \hat{\boldsymbol{\lambda}}(k)^T(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)$ is generated. With a new FC added into $\boldsymbol{C}_{11}$ (or a new IC added into $\boldsymbol{C}_{12}$) at the $(k+1)^{th}$ iteration, the MP is solved again to obtain a new solution $\boldsymbol{x}(k+1)$ for the next iteration.

3) Theoretical analysis:

Theorem 5.1: The non-optimal and infeasible values of the binary variables $\boldsymbol{x}$ are excluded by the constraints $\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{12}$.

Proof: In Case 2, since 1) $\hat{\Phi}(k) < \boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x}(k)-\boldsymbol{b}_2)$ due to *Lemma 4.1*, where $\hat{\Phi}(k)$ is the optimal solution of MP at the k^{th} iteration, and 2) $\boldsymbol{x}(k)$ is not the optimal solution of PP2 due to $\Phi_u(k)-\Phi_l(k)>\varepsilon$, the non-optimal solution $\boldsymbol{x}(k)$ is excluded by the constraint $\hat{\Phi} \geq \boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)$. In Case 3, the SP has no feasible solution under the given solution $\boldsymbol{x}(k)$. This problem is feasible if the positive variables $\boldsymbol{\xi} \triangleq [\xi_i]$ are introduced to relax the constraints. We construct a Feasibility Check Problem (FCP) and solve its dual problem (DFCP):

$$\mathbf{FCP}: \min_{\boldsymbol{y},\boldsymbol{\xi}\succeq 0} \ \mathbf{1}^T\boldsymbol{\xi} \quad \text{s.t.} \quad \boldsymbol{C}\boldsymbol{x}(k)+\boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2+\boldsymbol{\xi}, \tag{7}$$

$$\mathbf{DFCP}: \max_{\hat{\boldsymbol{\lambda}}\succeq 0} \ \hat{\boldsymbol{\lambda}}^T(\boldsymbol{C}\boldsymbol{x}(k)-\boldsymbol{b}_2) \quad \text{s.t.} \quad \mathbf{1}-\hat{\boldsymbol{\lambda}}\succeq 0, \boldsymbol{D}^T\hat{\boldsymbol{\lambda}}\succeq 0, \tag{8}$$

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

Lemma 5.2: The lower bound sequence $\{\Phi_l(k)\}$ is increasing and the upper bound sequence $\{\Phi_u(k)\}$ is decreasing.

Theorem 5.2: At each iteration with a new FC or IC added into the MP, the solution obtained by ODA converges to the global optimal value Φ^* within a finite number of iterations.

Proof: Based on *Theorem* 5.1 and *Lemma* 5.2, as well as the fact that the dimension of binary variables $\boldsymbol{x}$ is finite, the solution converges to the global optimal value within a finite number of iterations. ■

Remark 5.2: The reasons why DSP is solved instead of SP are that 1) the two problems are equivalent due to the strong duality, and 2) the FCs and the ICs are constructed by the solution of the DSP. The reasons why DFCB is solved instead of FCB are that 1) the two problems are equivalent due to the strong duality, and 2) $\hat{\boldsymbol{\lambda}}^T(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)$ is a function with respect to variables $\boldsymbol{x}$, but not $\mathbf{1}^T\boldsymbol{\xi}$, i.e., $0 \geq \mathbf{1}^T\boldsymbol{\xi}$ is an invalid constraint for the MP.

VI. Heuristic Task Deployment Algorithm

Although the solution found by ODA is optimal, this method cannot be used to efficiently solve large problem instances. This is because: 1) the MP is an ILP, which is hard to solve directly compared to the LP-based SP, and 2) the number of MP constraints increases with the number of iterations, due to the addition of FCs and ICs. In order to circumvent these difficulties, we propose a heuristic method. The structure of PP/PP1 shows that 1) the problem has a large number of binary and continuous variables, and 2) these variables are highly coupled with each other. Thus, it is difficult to design a heuristic based on the traditional multi-step optimization methods, such as [4], [12]. These methods usually fix one type of variables to determine the other type of variables. Using this multi-step method, the problem in the latter steps maybe unfeasible due to the solution provided by the previous steps. In contrast, inspired by the structure of ODA, we propose a novel heuristic algorithm HDA to efficiently solve PP2.

Theorem 6.1: The FC and IC generated by solving the DSP with $\boldsymbol{x}'(k)$ do not exclude the optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of PP2, where $\boldsymbol{x}'(k)$ is an arbitrary feasible solution of the MP.

Proof: If the DSP has a bounded solution $\boldsymbol{\lambda}'(k)$ with $\boldsymbol{x}'(k)$, a new FC is generated, i.e., $\hat{\Phi} \geq \boldsymbol{\lambda}'(k)^T(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)$. On the other hand, if the DSP has an unbounded solution with $\boldsymbol{x}'(k)$, a new IC is generated, i.e., $0 \geq \hat{\boldsymbol{\lambda}}'(k)^T(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b}_2)$, where $\hat{\boldsymbol{\lambda}}'(k)$ is the solution of DFCP with $\boldsymbol{x}'(k)$. Next, we prove that the solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ does not violate the above FC and IC. If the DSP has a bounded solution $\boldsymbol{\lambda}'(k)$ with $\boldsymbol{x}'(k)$, suppose that $\boldsymbol{x}^*$ and $\Phi^*=\boldsymbol{f}^T\boldsymbol{y}^*$ violate the FC, i.e., $\Phi^* < \boldsymbol{\lambda}'(k)^T(\boldsymbol{C}\boldsymbol{x}^*-\boldsymbol{b}_2)$. This contradicts the fact that Φ^* is the optimal value of the objective function of the DSP with $\boldsymbol{x}^*$ (i.e., $\Phi^*=\max_{\boldsymbol{\lambda}\succeq 0}\boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x}^*-\boldsymbol{b}_2) \geq \boldsymbol{\lambda}'(k)^T(\boldsymbol{C}\boldsymbol{x}^*-\boldsymbol{b}_2)$. Hence, the FC will not exclude the optimal solution $\boldsymbol{x}^*$. If the DSP has an unbounded solution with $\boldsymbol{x}'(k)$, $\boldsymbol{x}'(k)$ is excluded by the IC. Since $\boldsymbol{x}^* \neq \boldsymbol{x}'(k)$, $\boldsymbol{x}^*$ does not violate the IC. ■

Since MP is an ILP, the feasible solution $\boldsymbol{x}'(k)$ can be found through heuristics, such as the Feasibility Pump (FP) method [17]. The HDA structure is similar to the ODA structure except that 1) the optimal MP solution $\boldsymbol{x}(k)$ is replaced by a feasible MP solution $\boldsymbol{x}'(k)$ during the MP and SP iteration, and 2) the iteration stops, when the SP has a bounded solution for first time (Fig. 2).

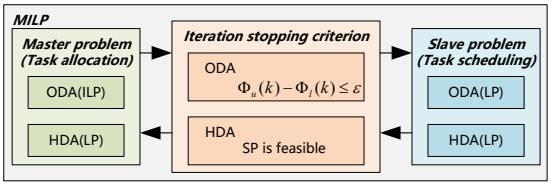


Fig. 2. The structure of ODA and HDA algorithms.

Definition 6.1: (*ϵ-optimal solution*) Denote $(\boldsymbol{x}, \boldsymbol{y})$ as the feasible solution of PP2, and $\Phi=\boldsymbol{f}^T\boldsymbol{y}$. If $|\Phi-\Phi^*| \leq \epsilon$, $(\boldsymbol{x}, \boldsymbol{y})$ is an ϵ-optimal solution.

Lemma 6.1: Denote $\Phi'_l(k)$ and $\Phi'_u(k)$ as the objective function values of the MP and the SP with $(\boldsymbol{x}'(k), \boldsymbol{y}'(k))$, respectively, where $(\boldsymbol{x}'(k), \boldsymbol{y}'(k))$ is the solution of PP2 found by HDA. The solution $(\boldsymbol{x}'(k), \boldsymbol{y}'(k))$ is an ϵ-optimal solution, where $\epsilon=\Phi'_u(k)-\Phi'_l(k)$.

TABLE II
FREQUENCY/VOLTAGE LEVELS AND POWER MODEL PARAMETERS OF A BIG CORE AND A LITTLE CORE

big										*LITTLE*					
v_k (V)	0.93	0.96	1.0	1.04	1.08	1.1	1.15	1.2	1.23	v_k (V)	0.9	0.94	1.01	1.09	1.2
f_k (GHz)	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6	f_k (GHz)	0.25	0.3	0.4	0.5	0.6
Power	C_k^s = 1.478, C_k^d = 0.471, ρ_k = 0.379									C_k^s = 1.191, C_k^d = 0.153, ρ_k = 0.757					

Compared with the existing heuristics, HDA is able to solve general MILP problem, even if the problem variables cannot be calculated separately. In addition, we know how far the feasible solution found by HDA from the optimal one.

VII. SIMULATION RESULTS

The AMP platform used in the experiments consists of four Cortex-A15 ("big") along with four Cortex-A7 ("LITTLE") cores [6]. Table II summarizes the AMP parameters obtained from [6]. The parameters of the experimental set-up are depicted in Table III. The ranges of the mandatory cycles M_l and the maximum optional cycles O_l are obtained from the work of [12], [18], where analytic experiments took place to measure the range in cycles for MiBench and MediaBench benchmarks. To set the task deadlines we introduce r_l and $\hat{t}_l^s$, i.e. the relative deadline and the temporary start time of task τ_l, respectively. If task τ_i precedes task τ_j and τ_j is the closest task of τ_i, the temporary end time of τ_i is set to the temporary start time of τ_j (i.e., $\hat{t}_i^e = \hat{t}_i^s + r_i = \hat{t}_j^s$). If τ_l is the first task in one hyper-period, $\hat{t}_l^s = 0$. $\underline{r}_l$ and $\overline{r}_l$ are the minimum and the maximum time required to execute the precise version of the tasks with $\{M_l + O_l\}$ $(\forall l \in \mathcal{N})$ cycles, respectively. E_h is the average minimum energy required to execute the precise version of the tasks with $\{M_l + O_l\}$ $(\forall l \in \mathcal{N})$ cycles. η is an energy efficiency factor. Not all the optional subtasks can be fully executed, i.e., $\{O_l\}$ $(\forall l \in \mathcal{N})$, as this violates the real-time and energy constraints. The QoS function of the tasks is adopted from [12], [13]. The use of arbitrary task graphs does not affect our problem formulation, as different task graphs only change the values of the problem parameters $\{\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{f}, \boldsymbol{b}_1, \boldsymbol{b}_2\}$. 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.

TABLE III
EXPERIMENTAL SET-UP

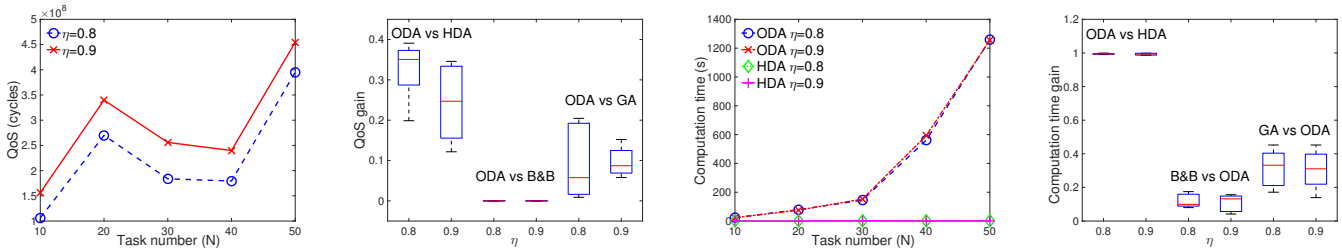
Task characteristics			
$M_l, O_l \in [4 \times 10^7, 6 \times 10^8]$		$d_l = \hat{t}_l^s + r_l$	
$H = \max_l\{d_l\}$		$r_l \in [\underline{r}_l, \overline{r}_l]$	
$\underline{r}_l = \min_k \left\{\frac{M_l+O_l}{f_k \lambda_{l,k}}\right\}$		$\overline{r}_l = \max_k \left\{\frac{M_l+O_l}{f_k \lambda_{l,k}}\right\}$	
Objective function		Energy supply	
$\sum_{l=1}^N \alpha_l o_l$, $0 \le \alpha_l \le 1$, $\sum_{l=1}^N \alpha_l = 1$		$E_s = \eta E_h$	
$E_h = \left[MH - \sum_{l=1}^N \left(\min_{\forall k} \left\{\frac{M_l+O_l}{f_k \lambda_{l,k}}\right\}\right)\right] \frac{\sum_{k=1}^M P_k^s}{M}$ $+ \sum_{l=1}^N \left[\min_{\forall k} \left\{\frac{M_l+O_l}{f_k \lambda_{l,k}} \left(P_k^s + P_k^d\right)\right\}\right]$			
Fixed parameters		Tuned parameters	
R	M	Min/Max/Step	N	η
2	8		10/50/10	0.8/0.9/0.1

Initially, we compare the performance (QoS and computation time) of the ODA and the HDA with: 1) optimal approach – Branch and Bound method (B&B) [19], known to provide optimal solution for the MILP problem, and 2) stochastic approach – Genetic Algorithm (GA), which is provided by the MATLAB optimization toolbox.

The QoS achieved by ODA with parameters N and η varying is shown in Fig. 3(a). We observe that 1) the QoS increases with η under the given N as more optional cycles can be executed, and 2) the QoS is not always increasing with N under the given η, because when N is changed, a new set of task parameters is generated. We further compare the QoS gain between ODA, HDA, B&B and GA in Fig. 3(b). The QoS gain "ODA vs HDA", "ODA vs B&B", and "ODA vs GA" shows the statistical property of data sets $\left\{\frac{Q_o(N,\eta)-Q_h(N,\eta)}{Q_o(N,\eta)}\right\}$, $\left\{\frac{Q_o(N,\eta)-Q_b(N,\eta)}{Q_o(N,\eta)}\right\}$, and $\left\{\frac{Q_o(N,\eta)-Q_g(N,\eta)}{Q_o(N,\eta)}\right\}$, where $Q_o(N,\eta)$, $Q_h(N,\eta)$, $Q_b(N,\eta)$ and $Q_g(N,\eta)$ are the QoS achieved by ODA, HDA, B&B and GA under the given N and η parameters, respectively. From Fig. 3(b), we observe that 1) the solutions given by B&B and ODA are same, and 2) ODA achieves higher QoS than HDA (29.8% in average) and GA (7.6% in average).

The computation time of ODA and HDA with parameters N and η varying is presented in Fig. 3(c). We observe that 1) η does not change the problem size, and, thus, its influence on computation time is limited, and 2) the influence of parameter N on the computation time of HDA is small, compared with ODA, and 3) the computation time of HDA is also influenced by using FP to solve the MP, since we need to repeatedly run FP until an arbitrary feasible solution is found. Fig. 3(d) shows the computation time gain between ODA, HDA, B&B and GA. Similarly, the computation time gain "ODA vs HDA", "B&B vs ODA", and "GA vs ODA" shows the statistical property of data sets $\left\{\frac{T_o(N,\eta)-T_h(N,\eta)}{T_o(N,\eta)}\right\}$, $\left\{\frac{T_b(N,\eta)-T_o(N,\eta)}{T_b(N,\eta)}\right\}$, and $\left\{\frac{T_g(N,\eta)-T_o(N,\eta)}{T_g(N,\eta)}\right\}$, where $T_o(N,\eta)$, $T_h(N,\eta)$, $T_b(N,\eta)$ and $T_g(N,\eta)$ are the computation time of ODA, HDA, B&B and GA under the given N and η parameters, respectively. With N increasing, the computation time of ODA and B&B grows. However, ODA takes a shorter computation time than B&B (16.3% in average). For an optimization problem, its computational complexity increases significantly with the number of variables and constraints. Hence, solving iteratively smaller problems with less variables and constraints (MP and SP) is more efficient than solving a single large problem. This result is in line with the comparison in [20]. Although GA can solve non-linear programming problem, e.g., MINLP, the optimality of the solution is hard to guarantee. Compared with ODA, the GA structure is more complex, and, thus, GA has a longer computation time (32.6% in average). The problem transformation from MINLP-based PP to MILP-based PP1 is necessary, as it can simplify the problem structure, and, thus, the optimal solution is much easier to find.

We further explore the convergence of our approaches by providing the number of iterations required to find the optimal (feasible) solution for ODA (HDA) in Fig. 4(a). For ODA more constraints are added to the problem with N increasing, and, thus, a higher number of iterations is required to find the optimal solution. However, for HDA, the number of required iterations

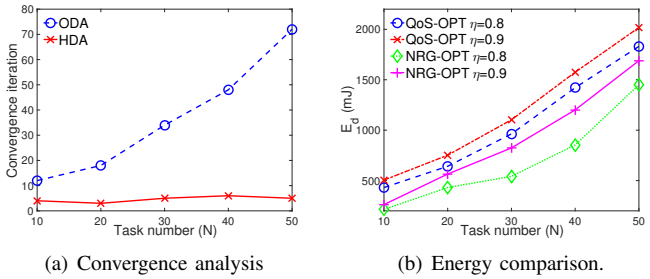


(a) QoS: ODA (b) QoS Comparison (c) Comp. time: ODA and HDA (d) Computation time comparison

Fig. 3. QoS (a)-(b) and computation time (c)-(d) achieved by the different methods with N and η varying.

is not always increasing with N. This is because 1) the initial solution $\boldsymbol{x}(0)$ is randomly selected as long as it satisfies the constraint $\boldsymbol{A}\boldsymbol{x}(0) \preceq \boldsymbol{b}_1$, and 2) the solution found by HDA, i.e., $\boldsymbol{x}'(k)$, is an arbitrary feasible solution.

Finally, we explore the behavior of our method by using ODA to solve the PP1 (QoS-OPT) and the energy-aware task mapping problem (NRG-OPT). The NRG-OPT aims to minimize the objective function $\sum_{k\in\mathcal{M}}[P_k^c t_k^d + (H - t_k^d)P_k^s]$ (total energy consumption) under the constraints $\boldsymbol{C}_1 - \boldsymbol{C}_6$ and $\boldsymbol{C}_8 - \boldsymbol{C}_{10}$. The results show that when the NRG-OPT is used for IC-task deployment, the QoS is always equal to 0. This behavior is expected as the smaller the optional subtask, the lower the energy consumed to execute it. Fig. 4(b) shows that QoS-OPT consumes more energy than NRG-OPT, since QoS-OPT maximizes QoS and, thus, executes more optional subtasks than NRG-OPT. However, the energy consumed by QoS-OPT is always less than the supplied energy E_s to satisfy the energy supply constraint. Hence, using QoS-OPT to perform IC-task deployment, we can balance QoS-enhancing with energy-usage.



(a) Convergence analysis (b) Energy comparison.

Fig. 4. Further exploration of the ODA and HDA behavior

VIII. Conclusions

This study formulated the approximation-aware task deployment problem on AMP platforms as an MILP, with a goal of maximizing QoS without violating the timing and the energy constraints. The problem is optimally solved by the proposed ODA algorithm. A novel heuristic algorithm HDA is also presented to further reduce the computation time. The numerical results show that ODA is guaranteed to converge to the optimal solution. It can reduce computation time by up to 46.8% and improve QoS by up to 21.1% compared with the existing approaches. HDA can find a feasible solution within a negligible computation time with an average cost of 29.8% in QoS compared with ODA.

Acknowledgment

This research is funded by INRIA post-doctoral research fellowship program, and is partially funded by ANR ARTE-FACT (AppRoximaTivE Flexible Circuits and Computing for IoT) project (Grant No. ANR-15-CE25-0015).

References

[1] S. Mittal, "A survey of techniques for architecting and managing asymmetric multicore processors," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 45:1–45:38, 2016.

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

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

[4] 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.

[5] 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.

[6] H. S. Chwa, J. Seo, H. Yoo, J. Lee, and I. Shin, "Energy and feasibility optimal scheduling on big.LITTLE platforms," in *Proc. IEEE RTSOPS*, 2013, pp. 770–777.

[7] 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.

[8] 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, pp. 1–17, 2017.

[9] 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.

[10] 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.

[11] L. Mo, A. Kritikakou, and O. Sentieys, "Controllable QoS for imprecise computation tasks on DVFS multicores with time and energy constraints," *IEEE J. Emerg. Sel. Topic Circuits Syst.*, pp. 1–14, 2018.

[12] 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/ACM DATE*, 2017, pp. 1402–1407.

[13] L. Mo, A. Kritikakou, and O. Sentieys, "Energy-quality-time optimized task mapping on DVFS-enabled multicores," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2428–2439, 2018.

[14] 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. 37, no. 9, pp. 1733–1746, 2018.

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

[16] 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.

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

[18] R. Jayaseelan and T. Mitra, "Temperature aware task sequencing and voltage scaling," in *Proc. IEEE ICCAD*, 2008, pp. 618–623.

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

[20] 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.