HAL open science

Optimal IC Task Mapping to Maximize QoS on Heterogeneous Multicore Systems

Lei Mo, Xinmei Li, Angeliki Kritikakou, Pengcheng You

To cite this version:

Lei Mo, Xinmei Li, Angeliki Kritikakou, Pengcheng You. Optimal IC Task Mapping to Maximize QoS on Heterogeneous Multicore Systems. IEEE Transactions on Circuits and Systems II: Express Briefs, 2023, pp.1-5. 10.1109/TCSII.2023.3329050 . hal-04267659

HAL Id: hal-04267659

https://hal.science/hal-04267659v1

Submitted on 3 Nov 2023

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

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

Distributed under a Creative Commons Attribution 4.0 International License

Optimal IC Task Mapping to Maximize QoS on Heterogeneous Multicore Systems

Lei Mo*, *Member, IEEE,* Xinmei Li*, *Student Member, IEEE,* Angeliki Kritikakou†, *Member, IEEE,* and Pengcheng You‡, *Member, IEEE*

***Abstract*—Heterogeneous multicore architectures have become one of the most widely used hardware platforms for embedded systems, where time, energy, and system QoS are the major concerns. The Imprecise Computation (IC) model splits a task into mandatory and optional parts, allowing the trade-off of the above issues. However, existing approaches, to maximize system QoS (Quality-of-Service) under time or energy constraints, use a linear function to model system QoS. Therefore, they become unsuitable for general applications, whose QoS is modeled by a concave function. To deal with this limitation, this work addresses the Mixed-Integer Non-Linear Programming (MINLP) problem of mapping IC tasks to a set of heterogeneous cores by concurrently deciding which processor executes each task and the number of cycles of optional tasks (i.e., task allocation and scheduling), under real-time and energy supply constraints. Furthermore, as existing solution algorithms either demand high time complexity or only achieve feasible solutions, we propose a novel approach based on problem transformation and dual decomposition that finds an optimal solution while avoiding high computational complexity. Simulation results show that the proposed approach achieves 98% performance of the optimization solver Gurobi, but only with 19.8% of its computation time.**

***Index Terms*—Task mapping, QoS, heterogeneous multicore, imprecise computation, dual decomposition.**

I. Introduction

With the increasing requirements for high performance and low task execution delay, multicore platforms have been widely used in various domains, such as Cyber-Physical Systems (CPS). The use of different types of processors in the same platform enables the specialization of the platform into the needs of the application domains, achieving parallel and efficient execution [1]. *Energy efficiency* and *real-time execution* are typically required during the embedded system design because 1) embedded systems have energy constraints, especially when they are powered by the battery, which has limited energy capacity, and 2) real-time responsiveness is required by many critical applications (e.g., target tracking or fire detection) since missing the application's deadline would lead to serious or even disastrous consequences. In several application domains, such as image processing, robot control, and information gathering, a task can be logically decomposed into a mandatory subtask and an optional subtask [2]. The mandatory subtask must be completed before the deadline, providing a baseline QoS. When it is executed, the optional subtask can increase the QoS, but it is not necessary to be executed completely. Such applications can be modeled by the IC tasks [3]. The way of mapping the IC tasks onto the hardware platforms plays an important role in the final system QoS. This is because the constraints on *energy consumption*, *real-time execution* and *system QoS* often conflict with each other. The longer the optional subtasks are executed, the higher QoS is achieved, while more energy and time are required.

Existing energy-aware task mappings approaches, e.g., [4], [5] aim to minimize energy consumption and typically use a precise task model. Therefore, they do not explore any QoS improvement by adjusting the optional cycles. On the contrary, the QoS-aware task mapping approaches [2], [3], [6]–[10] use the IC task model to maximize system QoS under real-time and/or energy constraints. For modeling the system QoS, the most realistic approaches are based on the linear function [2], [3], [6], [7] and the concave function [8]–[10]. The concave function is more general since it can characterize more extensive applications than the linear function [11]. However, existing QoS-aware approaches that model QoS through concave functions focus on either uni-processor platforms [10] or with fixed task-to-processor allocation [8] and omit energy-related issues [9]. To address this limitation, we propose a QoS-aware mapping problem for IC tasks with the general concave function on heterogeneous multicore platforms under energy consumption and real-time constraints. Then, we propose a novel polynomial time optimal approach to solve this complex MINLP problem efficiently.

II. System Model and Problem Formulation

1) System Model: We consider a set of periodic and independent IC tasks $\{\tau_1, \ldots, \tau_N\}$. They are released at time 0 and share a common hyper-period H [3], [12]. Each IC task τ_i contains a mandatory subtask with M_i cycles and an optional subtask with o_i cycles, where M_i and o_i are measured in Worst-Case Execution Cycles (WCEC). The optional subtask of τ_i is bounded by $0 \leq o_i \leq O_i$, where O_i is the number of maximum optional cycles. The values of M_i and O_i are determined by the minimum and maximum QoS [13].

*L. Mo and X. Li are with the Key Laboratory of Measurement and Control of CSE, Ministry of Education, School of Automation, Southeast University, 210096 Nanjing, China. E-mails: lmo@seu.edu.cn, xinmeili@seu.edu.cn.

†A. Kritikakou is with the University of Rennes, INRIA, IRISA, CNRS, 35042 Rennes, France, and is also with the Institut Universitaire de France, 75005 Paris, France. E-mail: angeliki.kritikakou@irisa.fr.

‡P. You is with the Department of Industrial Engineering and Management, College of Engineering, Peking University, 100871 Beijing, China. E-mail: pcyou@pku.edu.cn.

This research was partly supported by the National Key Research and Development Program of China (Grant No. 2022YFF0902800), the Southeast University "Zhishan Scholars" Project (Grant No. 2242021R40003), and the National Natural Science Foundation of China (Grant No. 61973163, 72201007, 72131001, and T2121002).

We consider a multicore platform with M heterogeneous processors $\{\theta_1, \ldots, \theta_M\}$. Each processor θ_k can operate at a voltage/frequency pair (v_k, f_k). We introduce a parameter $\lambda_{ik} \in (0, 1]$ [5] to describe the execution efficiency of processor θ_k, when θ_k executes task τ_i. Hence, the Worst-Case Execution Time (WCET) of task τ_i, when it is executed at frequency f_k on processor θ_k, is given by $\frac{M_i+o_i}{\lambda_{ik} f_k}$. The processor energy model is adopted from [12], where the power consumption of a processor θ_k is expressed as $P_k^c = P_k^s + P_k^d$. P_k^s is the static power of the processor ready to execute, and P_k^d is the dynamic power of task execution.

2) Problem Formulation: Given a set of N IC tasks, the goal is to map these tasks on M heterogeneous processors to maximize the overall system QoS under real-time and energy supply constraints. We consider a concave QoS function $g(\boldsymbol{o})$. More precisely, we determine 1) which processor the subtask should be executed on (*task allocation*), and 2) how many optional cycles a task should execute (*task scheduling*). To formulate the task mapping problem, we introduce the following variables: 1) define $\boldsymbol{s} \triangleq [s_{ik}]_{N\times M}$ and let binary variable $s_{ik} = 1$, if task τ_i is assigned to processor θ_k, otherwise, $s_{ik} = 0$; and 2) define $\boldsymbol{o} \triangleq [o_i]_{N\times 1}$ and let continuous variable o_i be the optional cycles of task τ_i. Taking time and energy constraints into account, the Primal Problem (PP) is given by

$$\begin{aligned} \textbf{PP}: \min_{\boldsymbol{s},\boldsymbol{o}} & -g(\boldsymbol{o}) \\ \text{s.t. } & \boldsymbol{C}_1 - \boldsymbol{C}_3,\ s_{ik} \in \{0,1\},\ 0 \le o_i \le O_i. \end{aligned} \tag{1}$$

Since each task τ_i is executed on only one processor, we obtain the following task allocation constraint $\boldsymbol{C}_1 : \sum_{k\in\mathcal{M}} s_{ik} = 1,\ \forall i \in \mathcal{N}$, where $\mathcal{N} \triangleq \{1, \ldots, N\}$. Considering task allocation decision s_{ik} and task execution cycles $M_i + o_i$, the time spent by processor θ_k to execute all the assigned tasks is known. Thus, the real-time constraint, which restricts all tasks assigned to processor θ_k being executed before the common deadline D, is given by $\boldsymbol{C}_2 : \sum_{i\in\mathcal{N}}(s_{ik}\frac{M_i+o_i}{\lambda_{ik} f_k}) \le D$, $\forall k \in \mathcal{M}$, where $\mathcal{M} \triangleq \{1, \ldots, M\}$. Note that the total energy consumed by M processors during the hyper-period H cannot exceed the energy supply E_s. The energy consumption is bounded by $\boldsymbol{C}_3 : \sum_{k\in\mathcal{M}}(T_k P_k^d + HP_k^s) = \sum_{k\in\mathcal{M}}[\sum_{i\in\mathcal{N}}(s_{ik}\frac{M_i+o_i}{\lambda_{ik} f_k})P_k^d + HP_k^s] \le E_s$.

Remark 2.1: We consider optional cycles o_i as a continuous variable for tractability reasons. When PP is solved, we round the result down. This impact of one cycle is negligible since tasks usually execute hundreds of thousands of cycles [14].

Remark 2.2: Since binary variable s_{ik} and continuous variable o_i are coupled with each other nonlinearly in $\boldsymbol{C}_2$ and $\boldsymbol{C}_3$, and the objective function is concave, PP is an MINLP.

Theorem 2.1: Task mapping problem (1) is $\mathcal{NP}$-hard [15].

Due to the page limit, the proofs are omitted here. Although nonlinear item $s_{ik}o_i$ can be linearized by introducing auxiliary variables and adding additional constraints into PP [4], this approach also increases problem size. To circumvent this difficulty, we replace nonlinear item $s_{ik}o_i$ with a new variable t_{ik}, representing the execution time of optional subtask o_i on processor θ_k. Using s_{ik} and t_{ik}, PP can be reformulated as

$$\begin{aligned} \textbf{PP1}: \min_{\boldsymbol{s},\boldsymbol{t}} & -g(\boldsymbol{t}) \\ \text{s.t. } & \boldsymbol{C}_1,\ \boldsymbol{C'}_2,\ \boldsymbol{C'}_3,\ \boldsymbol{C}_4,\ s_{ik} \in \{0,1\},\ t_{ik} \ge 0, \end{aligned} \tag{2}$$

where $\boldsymbol{C'}_2 : \sum_{i\in\mathcal{N}}(s_{ik}\frac{M_i}{\lambda_{ik} f_k} + t_{ik}) \le D$, $\forall k \in \mathcal{M}$ and $\boldsymbol{C'}_3 : \sum_{k\in\mathcal{M}}[\sum_{i\in\mathcal{N}}(s_{ik}\frac{M_i}{\lambda_{ik} f_k} + t_{ik})P_k^d + HP_k^s] \le E_s$. With $\boldsymbol{C}_4 : t_{ik} \le s_{ik}\frac{O_i}{\lambda_{ik} f_k}$, $\forall i \in \mathcal{N}$, $\forall k \in \mathcal{M}$, if a task τ_i is assigned to the processor θ_k, $s_{ik} = 1$. The task execution time of θ_k with respect to the optional subtask of τ_i is bounded by $0 \le t_{ik} \le \frac{O_i}{\lambda_{ik} f_k}$, else, $t_{ik} = 0$. Since s_{ik} and t_{ik} are coupled with each other linearly in $\boldsymbol{C}_1$–$\boldsymbol{C}_4$, (2) is an MILP problem.

Lemma 2.1: Mapping problems (1) and (2) are equivalent.

III. Optimal QoS-aware Task Mapping Method

This section proposes a low computational complexity algorithm to optimally solve the PP1 according to the problem structure. Due to binary variables $\boldsymbol{s}$, PP1 is a non-convex problem, making this problem difficult to solve. To reduce the computational complexity, we can relax $\boldsymbol{s}$ from binary to continuous variables. However, the feasible region of variables $\boldsymbol{s}$ in PP1 is also changed. Nevertheless, we can achieve an equivalent transformation if we solve the relaxed problem properly, following the three steps below.

1) Problem Relaxation: We relax the binary variable s_{ik} to a continuous variable within the range $[0, 1]$. Therefore, (2) is transformed into the following problem:

$$\begin{aligned} \textbf{PP2}: \min_{\boldsymbol{s},\boldsymbol{t}} & -g(\boldsymbol{t}) \\ \text{s.t. } & \boldsymbol{C}_1 - \boldsymbol{C}_4,\ 0 \le s_{ik} \le 1,\ t_{ik} \ge 0. \end{aligned} \tag{3}$$

Let Φ_1^* and Φ_2^* be the optimal objective function values of PP1 and PP2, respectively. Compared with PP1, the feasible region is enlarged in PP2. Since PP1 and PP2 are minimization problems, Φ_1^* is actually an upper bound of Φ_2^*, i.e., $\Phi_1^* \ge \Phi_2^*$.

Theorem 3.1: The relaxed problem (3) is convex.

Note that the convex problem (3) can be optimally solved by the polynomial-time method. If the optimal solution to the PP2 also satisfies the constraints in the PP1, we get $\Phi_1^* = \Phi_2^*$. To find this type of solution, the constraint $s_{ik} \in \{0, 1\}$ can be combined into the solving process of PP2. Here, we explain how to solve PP1 based on the solution of PP2.

2) Dual Problem Construction and Refinement: Instead of solving PP2 directly, we solve its dual problem, as PP2 is a convex problem. To construct the dual problem, we introduce Lagrange multipliers $\boldsymbol{\alpha}$, $\boldsymbol{\beta} \succeq 0$, $\boldsymbol{\gamma} \succeq 0$ and $\mu \ge 0$ with proper dimensions to its constraints. Hence, the Lagrangian is $\mathcal{L}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mu) = -g(\boldsymbol{t}) + \sum_{i\in\mathcal{N}}(\sum_{k\in\mathcal{M}} s_{ik} - 1)\alpha_i + \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{M}}(t_{ik} - s_{ik}\frac{O_i}{\lambda_{ik} f_k})\beta_{ik} + \sum_{k\in\mathcal{M}}[\sum_{i\in\mathcal{N}}(s_{ik}\frac{M_i}{\lambda_{ik} f_k} + t_{ik}) - D]\gamma_k + \{\sum_{k\in\mathcal{M}}[\sum_{i\in\mathcal{N}}(s_{ik}\frac{M_i}{\lambda_{ik} f_k} + t_{ik})P_k^d + HP_k^s] - E_s\}\mu$. The dual function $\mathcal{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mu)$ is defined as the minimum value of the Lagrangian $\mathcal{L}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mu)$ over the variables $\boldsymbol{s}$ and $\boldsymbol{t}$ [16]. Thus, for the given $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and μ, we have objective function $\mathcal{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mu) = \min_{\boldsymbol{s},\boldsymbol{t}} \mathcal{L}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mu)$, and the Dual Problem (DP) associated with PP2 is given by

$$\begin{aligned} \textbf{DP}: \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu} & \mathcal{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mu) \\ \text{s.t. } & \boldsymbol{\beta} \succeq 0,\ \boldsymbol{\gamma} \succeq 0,\ \mu \ge 0. \end{aligned} \tag{4}$$

From (3), we observe that the inequalities $\boldsymbol{C}_2$–$\boldsymbol{C}_4$ are affine, and thus, the Slater's condition [16] is satisfied.

Lemma 3.1: A strong duality exists between the primal problem (3) and the dual problem (4).

Due to the strong duality, the primal problem and the dual problem have the same optimal solution [16], i.e., solving problem (4) and problem (3) are equivalent. To solve the problem of maximizing $\mathcal{R}(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$, we first solve the problem of minimizing $\mathcal{L}(\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$. Since variables $\boldsymbol{s}$ and $\boldsymbol{t}$ are coupled with each linearly in $\mathcal{L}(\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$, the Lagrangian can be decomposed as $\mathcal{L}(\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) = \mathcal{L}_1(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) + \mathcal{L}_2(\boldsymbol{t},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) + \mathcal{L}_3(\boldsymbol{\gamma},\mu)$, where $\mathcal{L}_1(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) = \sum_{i\in\mathcal{N}}\{\sum_{k\in\mathcal{M}}[\alpha_i - \frac{O_i\beta_{ik}-M_i(\gamma_k+P_k^d\mu)}{\lambda_{ik}f_k}]s_{ik} - \alpha_i\}$, $\mathcal{L}_2(\boldsymbol{t},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) = \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{M}}(\beta_{ik}+\gamma_k+P_k^d\mu)t_{ik} - g(\boldsymbol{t})$, and $\mathcal{L}_3(\boldsymbol{\gamma},\mu) = \sum_{k\in\mathcal{M}}(HP_k^s\mu - D\gamma_k) - E_s\mu$. Note that $\mathcal{L}_1(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$ and $\mathcal{L}_2(\boldsymbol{t},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$ are the functions regarding variables $\boldsymbol{s}$ and $\boldsymbol{t}$, while $\mathcal{L}_3(\boldsymbol{\gamma},\mu)$ is fixed under the given multipliers $\boldsymbol{\gamma}$ and μ. The dual function in (4) is written as $\mathcal{R}(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) = \min_{\boldsymbol{s}}\mathcal{L}_1(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) + \min_{\boldsymbol{t}}\mathcal{L}_2(\boldsymbol{t},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) + \mathcal{L}_3(\boldsymbol{\gamma},\mu)$. Hence, $\min_{\boldsymbol{s}}\mathcal{L}_1(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$ and $\min_{\boldsymbol{t}}\mathcal{L}_2(\boldsymbol{t},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$ can be solved separately.

A) To minimize $\mathcal{L}_1(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$ by $\boldsymbol{s}$, let $\mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu) = \frac{O_i\beta_{ik}-M_i(\gamma_k+P_k^d\mu)}{\lambda_{ik}f_k}$ and substitute it into $\mathcal{L}_1(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$:

$$\begin{aligned}&\min_{\boldsymbol{s}}\mathcal{L}_1(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)\\ =&\min_{\boldsymbol{s}}\sum\nolimits_{i\in\mathcal{N}}\left\{\sum\nolimits_{k\in\mathcal{M}}[\alpha_i - \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)]s_{ik} - \alpha_i\right\}.\end{aligned} \tag{5}$$

Since $[\alpha_i - \mathcal{H}_{i,k}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)]s_{ik}$ is a linear function regarding the variable s_{ik}, (5) can be decomposed into $N\times M$ independent problems with the forms: $\min_{s_{ik}}(\alpha_i - \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu))s_{ik}$, $\forall i\in\mathcal{N}$, $\forall k\in\mathcal{M}$. Comparing α_i with $\mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu))$, the task allocation decision s_{ik} can be determined as follows:

$$s_{ik} = \begin{cases} 1 & \text{if } \alpha_i < \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)\\ [0,1] & \text{if } \alpha_i = \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)\\ 0 & \text{if } \alpha_i > \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)\end{cases} \tag{6}$$

From (6) we have $\min_{s_{ik}}[\alpha_i - \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)]s_{ik} = [\alpha_i - \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)]^-$, where $[x]^- \triangleq \min\{0,x\}$. The above equation shows that (5) can be decomposed into N independent functions with the forms: $\sum_{k\in\mathcal{M}}[\alpha_i - \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)]^-$, $\forall i\in\mathcal{N}$.

Lemma 3.2: Since each task τ_i is executed on only one processor, to minimize the function $\sum_{k\in\mathcal{M}}[\alpha_i - \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)]^-$, the optimal task allocation decision s_{ik}^* is determined by:

$$s_{ik}^* = \begin{cases} 1 & \text{if } k = \arg\min_k\{\alpha_i - \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)\}\\ 0 & \text{else}\end{cases} \tag{7}$$

If $\alpha_i - \mathcal{H}_{ik}(\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$ has multiple same minimum values, we randomly select one from these minimum items, as each task is assigned to only one processor.

B) Under the given multipliers $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and μ, in order to find an optimal task execution time $\boldsymbol{t}^*$ to minimize $\mathcal{L}_2(\boldsymbol{t},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$, we solve the following differential equations: $\frac{\partial\mathcal{L}_2(\boldsymbol{t},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)}{\partial\boldsymbol{t}} = 0 \Rightarrow \boldsymbol{t}$. Based on $\boldsymbol{C}_2$ in PP1, task execution time $\boldsymbol{t}^*$ is influenced by the task allocation decision $\boldsymbol{s}^*$. The optimal execution time of optional task t_{ik}^* can be selected as follows:

$$t_{ik}^* = \begin{cases} t_{ik} & \text{if } s_{ik}^* = 1\\ 0 & \text{else}\end{cases} \tag{8}$$

Algorithm 1: QoS-aware Task Mapping Algorithm

1 **Input**: Parameters in (3);
2 **Output**: $\boldsymbol{s}$ and $\boldsymbol{t}$;
3 **Initialize**: Lagrangian multipliers $\{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu\}$, $j=0$, $\delta_j=\delta$, ϵ;
4 **while** $j < N_{\max}$ **do**
5 Update $\boldsymbol{s}^{(j)}$ and $\boldsymbol{t}^{(j)}$ through (7) and (8);
6 Calculate $g^{(j)}(\boldsymbol{t})$ and $\mathcal{L}^{(j)}(\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$;
7 **if** $|\mathcal{L}(\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) - (-g(\boldsymbol{t}))| < \epsilon$ **then**
8 $\boldsymbol{s}^* = \boldsymbol{s}^{(j)}$ and $\boldsymbol{t}^* = \boldsymbol{t}^{(j)}$;
9 **end**
10 $\boldsymbol{x}^{(j+1)} \leftarrow [\boldsymbol{x}^{(j)} + \delta_j \bigtriangledown \boldsymbol{x}^{(j)}]^+$, where $\boldsymbol{x} = \boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu$;
11 Set $j \leftarrow j+1$, and update $\delta_{j+1} \leftarrow \delta_j/\sqrt{j}$;
12 **end**

Thus, we have $\min_{\boldsymbol{t}}\mathcal{L}_2(\boldsymbol{t},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) = \mathcal{L}_2(\boldsymbol{t}^*,\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$, where $\boldsymbol{t}^*$ and $\mathcal{L}_2(\boldsymbol{t}^*,\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$ are actually the functions with respect to multipliers $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and μ. Substituting (7) and (8) into the objective function of (4), the refined dual function is

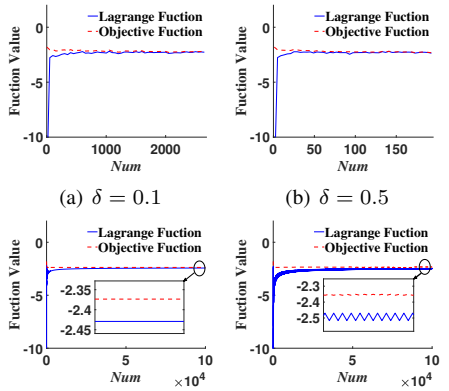
$$\mathcal{R}(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) = \mathcal{L}_1(\boldsymbol{s}^*,\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) + \mathcal{L}_2(\boldsymbol{t}^*,\boldsymbol{\beta},\boldsymbol{\gamma},\mu) + \mathcal{L}_3(\boldsymbol{\gamma},\mu). \tag{9}$$

On this basis, we replace the objective function in (4) with the refined dual function (9), and solve the refined dual problem by determining Lagrangian multipliers $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and μ.

3) Optimal Task Mapping Solution: To solve the refined dual problem, an important step is to find the subgradient of the dual function (9). Let $\bigtriangledown\alpha_i$, $\bigtriangledown\beta_{ik}$, $\bigtriangledown\gamma_k$ and $\bigtriangledown\mu$ denote the subgradient of dual function $\mathcal{R}(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$ with respect to the variables α_i, β_{ik}, γ_k and μ, respectively. Based on the definition of subgradient [17], we have $\bigtriangledown\alpha_i = \sum_{k\in\mathcal{M}}s_{i,k}^* - 1$, $\bigtriangledown\beta_{i,k} = t_{i,k}^* - s_{i,k}^*\frac{O_i}{\lambda_{i,k}f_k}$, $\bigtriangledown\gamma_k = \sum_{i\in\mathcal{N}}(s_{i,k}^*\frac{M_i}{\lambda_{i,k}f_k} + t_{i,k}^*) - D$, and $\bigtriangledown\mu = \sum_{k\in\mathcal{M}}[\sum_{i\in\mathcal{N}}(s_{i,k}^*\frac{M_i}{\lambda_{i,k}f_k} + t_{i,k}^*)P_k^d + HP_k^s] - E_s$, where s_{ik}^* and t_{ik}^* are corresponding optimal solutions of task allocation and execution time under the given multipliers $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and μ, respectively. Algorithm 1 summarizes the proposed optimal QoS-aware task mapping method. In each iteration, the Lagrange multipliers are updated with subgradient equations (Line 10), where $[x]^+ \triangleq \max\{0,x\}$. Then, the decisions regarding the task allocation and execution time are updated through (7) and (8) (Line 5). Finally, the iteration stops when $|\mathcal{L}(\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu) - (-g(\boldsymbol{t}))| \le \epsilon$, where $-g(\boldsymbol{t})$ is the objective function, $\mathcal{L}(\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mu)$ is the Lagrangian, and ϵ is a small positive value that controls the convergence of algorithm [17] (Lines 6–9). The refined dual problem can be solved in polynomial time with the above method. Assume that $\{\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*,\boldsymbol{\gamma}^*,\mu^*\}$ to be the global optimal solution to the refined dual problem. Since the multipliers are optimal, the corresponding solution $\{\boldsymbol{s}^*,\boldsymbol{t}^*\}$ is also optimal.

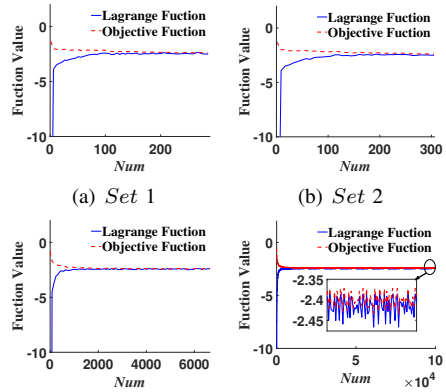
Remark 3.1: Transforming dual problem from (2) to (4), we avoid solving s_{ik} directly since s_{ik} can be expressed by other variables $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and μ. (7) and (8) show that by properly solving the problem of minimizing Lagrangian, we can cut the feasible region of s_{ik} and force s_{ik} to be a binary variable.

Remark 3.2: (6) and (7) show that s_{ik} can only be set to 0 or 1 when solving the PP2. Moreover, the objective function and the constraints of PP1 and PP2 are the same. Hence, the optimal solution to the PP2 also meets the constraints in the PP1, i.e., $\{\boldsymbol{s}^*,\boldsymbol{t}^*\}$ is also the optimal solution to the PP1.



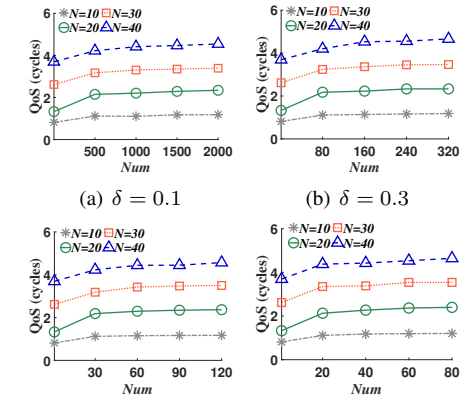
(a) $\delta = 0.1$ (b) $\delta = 0.5$ (c) $\delta = 1$ (d) $\delta = 5$

Fig. 1. Algorithm convergence under δ varying.



(a) Set 1 (b) Set 2 (c) Set 3 (d) Set 4

Fig. 2. Algorithm convergence under $\boldsymbol{\phi}^{(0)}$ varying.



(a) $\delta = 0.1$ (b) $\delta = 0.3$ (c) $\delta = 0.5$ (d) $\delta = 0.7$

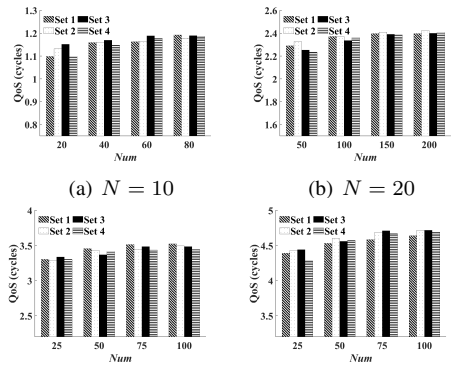
Fig. 3. QoS of different task sets under δ varying.

Remark 3.3: In each iteration, considering the number of variables, the time complexity is $O(3NM+N+M+1)$ [18]. In addition, we have $N_{\max} \gg M$ and $N \gg M$, and in the worst case, the maximum number of iterations is $N_{\max}$. Hence, the time complexity can be simplified as $O(N_{\max}N)$.

IV. Experimental Results

The simulations are based on a multi-core platform with six heterogeneous processors ($M = 6$). The processor parameters can be found in [4], [12], which are based on $70nm$ technology. The task efficiency factor is set to $\lambda_{ik} \in (0, 1]$ [5]. The cycles of mandatory subtasks M_i and maximum optional subtasks O_i are assumed within the range $[4 \times 10^7, 6 \times 10^8]$ [12]. This range is calculated from the MiBench and MediaBench benchmarks. To let time and energy constraints change with task number N, we set task deadline $D = H = \theta D_h$ and energy supply $E_s = \eta E_h$, where θ, $\eta \in (0, 1]$ are time and energy factors, $D_h = \sum_{i\in\mathcal{N}} \max_k\{\frac{M_i+O_i}{\lambda_{ik} f_k}\}$ and $E_h = \sum_{k\in\mathcal{M}} P_k^s D + \sum_{i\in\mathcal{N}} \max_k\{\frac{M_i+O_i}{\lambda_{ik} f_k} P_k^d\}$ are the maximum time and energy required to execute N tasks and each task τ_i has $M_i + O_i$ cycles. We adopt a concave QoS function $g(\boldsymbol{t}) = \sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{M}} (-p_{ik} t_{ik} + q_{ik}\sqrt{t_{ik}})$ from [10], where p_{ik} and $q_{ik} \in (0, 1]$ are coefficients related to applications. For Algorithm 1, we set stop criteria $\epsilon = 10^{-3}$, maximum iteration number $N_{\max} = 10^5$, and initial multipliers $\boldsymbol{\phi}^{(0)} = \{\boldsymbol{\alpha}^{(0)}, \boldsymbol{\beta}^{(0)}, \boldsymbol{\gamma}^{(0)}, \mu^{(0)}\} \in [0, 1]$. Although different processor platforms and real application tasks will lead to different PP parameters, the problem structures are still the same, i.e., the proposed method is applicable for different parameter values.

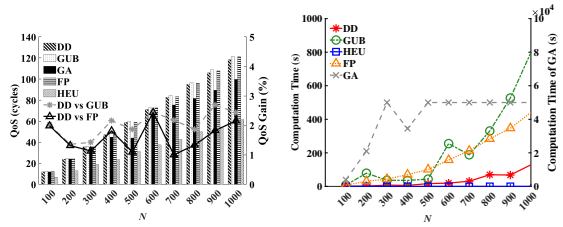
The values of θ and η mainly influence the feasibility of the problem since they change time and energy constraints. In contrast, the values of step-size δ and initial Lagrange multipliers $\boldsymbol{\phi}^{(0)}$ affect the solution quality and the iteration number of our approach when the problem is feasible. In Fig. 1, $N = 20$; we fix the initial Lagrange multipliers $\boldsymbol{\phi}^{(0)}$ and change the step-size δ from 0.1 to 5. The result shows that the algorithm convergence speed increases with δ, as a small step-size will increase the number of iterations and slow the convergence speed ($\delta = 0.1, 0.5$). However, a large step-size will miss the optimal solution ($\delta = 1$) or even lead to oscillation ($\delta = 5$). This result aligns with the stability analysis in [17]. The value of δ should be carefully determined based on the problem parameters. In Fig. 2, we set $N = 20$ and $\delta = 0.5$ and use different initial Lagrange multipliers. Set 1: $\boldsymbol{\phi}^{(0)} \in [0, 0.5]$, Set 2: $\boldsymbol{\phi}^{(0)} \in [0.5, 1]$, Set 3: $\boldsymbol{\phi}^{(0)} \in [1, 1.5]$, and Set 4: $\boldsymbol{\phi}^{(0)} \in [1.5, 2]$. When the values of $\boldsymbol{\phi}^{(0)}$ are large (Set 3 and Set 4), the convergence speed is slower, or the result is hard to converge (Set 4). However, when the results are stable (Set 1 and Set 2), the influence of $\boldsymbol{\phi}^{(0)}$'s value is limited, compared with δ, as the differences between these sets are usually within 100 iterations. To reduce computation time (iteration number), we can fix $\boldsymbol{\phi}^{(0)}$ and then adjust δ.



(a) $N = 10$ (b) $N = 20$ (c) $N = 30$ (d) $N = 40$

Fig. 4. QoS of different task sets under $\boldsymbol{\phi}^{(0)}$ varying.

Fig. 3 and Fig. 4 compare system QoS achieved by the proposed approach under different iteration numbers Num and task numbers N. In Fig. 3, we fix $\boldsymbol{\phi}^{(0)}$ and change δ, while in Fig. 4, we fix $\delta = 0.5$ and use different $\boldsymbol{\phi}^{(0)}$. Note that Fig. 1 and Fig. 2 show the results when the iterations are stopped (i.e., the convergence criteria are satisfied); Fig. 3 and Fig. 4 show the results achieved by a given iteration number. Since our approach is iteration-based, the more iterations are performed, the better the solution quality, as shown in Fig. 3. Our approach initially converges very fast. With the iteration number increasing, the improvement in solution quality is reduced. For instance, when $N = 10$ and $\delta = 0.1$, the QoS loss between $Num = 500$ and $Num = 2000$ is less than 5%. However, the computation time of $Num = 2000$ is four times of $Num = 500$. This is because the adaptive step-size method is used in Algorithm 1. The step-size is updated by $\delta_{j+1} \leftarrow \delta_j/\sqrt{j}$, i.e., δ_{j+1} decreases with iteration number j. In some time-sensitive applications, the iteration can be stopped when solution quality already reaches the requirements (e.g., the time and energy constraints are met or the gap between



(a) System QoS. (b) Computation time.

Fig. 5. QoS and computation time comparisons of different methods.

the achieved and optimal solutions is within a threshold), even if the convergence criteria are not satisfied. Fig. 4 shows that under the given N and Num parameters, the QoS differences between different initial multipliers $\boldsymbol{\phi}^{(0)}$ are small, as δ plays a more important role than $\boldsymbol{\phi}^{(0)}$ to influence solution quality.

Fig. 5 compares system performance (QoS and computation time) of the proposed approach, i.e., Dual Decomposition-based (DD) approach, with the state-of-the-art optimization solver Gurobi (GUB), Genetic Algorithm (GA), Feasibility Pump method (FP) [19] and a two-step heuristic (HEU) [12]. The Matlab optimization toolbox provides GA. The QoS gain between GUB and DD is defined as $\frac{f_{GUB}(N)-f_{DD}(N)}{f_{GUB}(N)}$, where $f_{GUB}(N)$ and $f_{DD}(N)$ is the QoS achieved by GUB and DD under the given N parameter. We set $\delta = 0.5$ and vary task number N from 100 to 1000, and each time $\boldsymbol{\phi}^{(0)}$ is randomly selected within $[0, 1]$. Fig. 5(a) shows that DD outperforms GA and HEU in terms of QoS improvement (7.8% and 48% on average). Although GA can solve complex mixed programming problems, such as MINLP and MILP, the solution's optimality is hard to guarantee. The HEU method, which determines task allocation and optional cycle adjustment in sequence, has a lower system QoS since separating task allocation and optional cycle adjustment reduces the feasibility region of the original problem. To enhance solution quality, the subproblems should be considered simultaneously. Fig. 5(b) compares the computation time of different methods. DD achieves about 98% and 98.3% (on average) performance of GUB and FP, but only with 19.8% and 14.4% computation time. This is because PP (1) is solved in a convex manner, and the computation time is less than solving an MINLP. The result shows that the computation time of DD almost grows linearly with task number N. This result aligns with the complexity analysis. Since our approach can transform MINLP into a convex problem equivalently, it achieves a balance between computation time and solution quality. Although DD and GA are both iteration-based methods, the GA structure is more complex than DD, as it generates new populations through several procedures at each iteration, e.g., selection, reproduction, mutation and crossover. Fig. 5 shows that when N is large, DD obtains a better QoS in a short time compared with GA. For the iteration approach, only when the step-size is small enough and the iteration number is large enough, we can reach the optimal solution [17]. The above results of DD can be further improved through better δ and Num parameters.

V. Conclusions

This paper studied the QoS-aware task mapping problem for IC tasks on heterogeneous multicore platforms to maximize the system QoS under real-time and energy constraints, formulated as an MINLP problem. Based on the problem transformation and dual decomposition, we proposed a novel method to efficiently solve this problem in low computation time, whose solution is the same as the optimal solution to the original problem. Finally, the simulation results show the effectiveness of the proposed method, which outperforms other methods in terms of QoS improvement and computation time.

References

[1] J. Zhou, Y. Shen, L. Li, C. Zhuo, and M. Chen, "Swarm intelligence-based task scheduling for enhancing security for IoT devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 6, pp. 1756–1769, 2023.

[2] S. Chakraborty, S. Saha, M. Själander, and K. Mcdonald-Maier, "PREPARE: Power-aware approximate real-time task scheduling for energy-adaptive QoS maximization," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5, pp. 1–25, 2021.

[3] L. Mo, A. Kritikakou, and O. Sentieys, "Decomposed task mapping to maximize QoS in energy-constrained real-time multicores," in *IEEE International Conference on Computer Design*, 2017, pp. 493–500.

[4] 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.

[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] M. Ansari, S. Safari, S. Yari-Karin, P. Gohari-Nazari, H. Khdr, M. Shafique, J. Henkel, and A. Ejlali, "Thermal-aware standby-sparing technique on heterogeneous real-time embedded systems," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 4, pp. 1883–1897, 2022.

[7] K. Cao, J. Zhou, G. Xu, T. Wei, and S. Hu, "Exploring renewable-adaptive computation offloading for hierarchical QoS optimization in fog computing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2095–2108, 2020.

[8] H. Yu, Y. Ha, B. Veeravalli, F. Chen, and H. El-Sayed, "DVFS-based quality maximization for adaptive applications with diminishing return," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 803–816, 2021.

[9] G. M. Tchamgoue, K. H. Kim, Y. K. Jun, and W. Y. Lee, "Compositional real-time scheduling framework for periodic reward-based task model," *Journal of Systems and Software*, vol. 86, no. 6, pp. 1712–1724, 2013.

[10] 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.

[11] 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.

[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. Design, Automation and Test in Europe*, 2017, pp. 1402–1407.

[13] 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.

[14] 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.

[15] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: a survey," *Surveys in Operations Research and Management Science*, vol. 17, no. 2, pp. 97–106, 2012.

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

[17] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *Lecture notes of EE392o, Stanford University*, pp. 2004–2005, 2003.

[18] O. He, S. Dong, W. Jang, J. Bian, and D. Z. Pan, "UNISM: Unified scheduling and mapping for general networks on chip," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 8, pp. 1496–1509, 2012.

[19] D. E. Bernal, S. Vigerske, F. Trespalacios, and I. E. Grossmann, "Improving the performance of DICOPT in convex MINLP problems using a feasibility pump," *Optimization Methods and Software*, vol. 35, no. 1, pp. 171–190, 2020.