Real-Time Scheduling of Periodic Tasks with Hard Deadlines: A Multi-Objective Approach for Energy and Performance Trade-offs

***Abstract*—**
***Index Terms*—**

I. Introduction

The rapid evolution of real-time embedded systems in safety-critical domains—such as automotive, aerospace, and healthcare—demands robust task scheduling frameworks that guarantee hard real-time constraints while optimizing secondary objectives like energy efficiency and performance. In many modern applications, periodic tasks are composed of both mandatory and optional components. The mandatory part must meet strict timing guarantees to ensure safety and functional correctness, whereas the optional part enhances performance or system utility when resources permit.

A representative example is the Adaptive Cruise Control (ACC) system in autonomous and semi-autonomous vehicles. ACC continuously monitors the road using radar, lidar, and camera sensors to maintain safe inter-vehicle distances and adapt vehicle speed. This functionality can be decomposed into periodic tasks, each with different levels of criticality and computational demand. Consider the following two tasks:

- **Task τ_1: Obstacle Detection and Avoidance.** This task is released every 100 ms and includes a mandatory component ($C_1^m = 20$ ms) that performs radar-based object detection and an optional component ($C_1^o = 15$ ms) for fusing camera data to enhance object classification accuracy. The optional part, while not safety-critical, improves situational awareness and decision-making.
- **Task τ_2: Speed Profile Adjustment.** Also periodic with a 100 ms period, this task includes a mandatory PID control loop ($C_2^m = 10$ ms) to maintain distance and an optional Model Predictive Control (MPC) module ($C_2^o = 10$ ms) to achieve smoother acceleration and fuel efficiency.

These tasks illustrate the dual objectives in modern automotive systems: guaranteeing safety through timely execution of mandatory jobs and enhancing performance and efficiency via optional jobs—all under limited computational and energy budgets.

II. Decision Variables

- $y_{ix} \in \{0, 1\}$: 1 if optional execution of J_{ix}^o is performed.
- $z_{ix}^{jf} \in \{0, 1\}$: 1 if job J_{ix} is assigned to processor P_j at frequency f; 0 otherwise.

III. System Model and Problem Formulation

A. System Model

In this section, we present the system model and formulate the problem being addressed. Various notations used throughout this paper are listed in Table 1.

Processor Model: Let $\{P_j\}$ be a set of h heterogeneous DVFS enabled processors (or cores) such that the j^{th} processor[1] P_j can execute with a discrete set of f frequency levels $\{F_j^k\}|F_j^k < F_j^{k+1}$. Thus, F_j^1 (and F_j^f) represents the minimum (and maximum) frequency of P_j. Without loss of generality, we assume that the frequency values are normalized, and thus, $F_j^f \leq 1.0$.

Task Model: Let $\{T_i\}$ be a set of n non-preemptive hard real-time tasks. Each task T_i is characterized by the parameters $\langle c_i^m, c_i^o, d_i \rangle$ where c_i^m, c_i^o, and p_i $(\geq c_i^m + c_i^o)$ represent the worst-case mandatory execution time must finish before deadline, worst case optional execution time can be executed if resource permit, and period (as well as relative deadline) of task T_i, respectively.

The x^{th} activation of T_i (henceforth, referred to as *job* J_{ix}) arrives at $(x-1)p_i$ with relative deadline xp_i. As it is hard real-time tasks all jobs of real-time tasks are equally important, and at lest mandatory execution of the task must complete within the deadline.

The hyper-period H is computed as $H = \{p_i\}$. Moreover, let n_i denotes the number of jobs of T_i in the hyper-period H, and η is the total number of jobs of all tasks.

$$n_i = \frac{H}{p_i} \Rightarrow \eta = \sum_i n_i$$

Let f_{ix} denote the frequency with which job J_{ix} is scheduled. Thus, the effective execution time of J_{ix} is given as $\frac{e_i}{f_{ix}}$.

Power Model: A DVFS-enabled processor is in the active state (running with one of the discrete frequency levels) or in the sleep state [**?**], [**?**], [**?**], [**?**]. We assume that static power is always consumed and focus on optimizing dynamic power which has two components: a frequency-dependent and a frequency-independent components. The frequency scaling beyond the *energy efficient frequency* $\upsilon \approx \left(\frac{\kappa}{2h}\right)^{\frac{1}{3}}$ may result in

[1]For the sake of uniformity, we use i, x, j, and k to index the set of tasks, jobs, processors, and frequencies, respectively. Furthermore, in this paper, for example, $\sum_j$ is used in place of $\sum_{j=1}^{h}$.

a larger energy consumption due to longer frequency independent power consumption [**?**], [**?**]. For the sake of simplicity, we assume that the smallest frequency on any processor $F_j^1 \geq \upsilon$ and focus on optimizing energy consumption determined by frequency dependent component only. The energy consumption to execute T_i^x is

$$E_i^x = (f_{ix})^2 \times (c_i^m + y_{ix} \times c_i^o) \tag{1}$$

The energy consumption to execute all jobs of T_i in the hyper-period H is $E_i = \sum_x E_i^x$. Thus, the system level energy consumption is

$$E = \sum_i \sum_x E_i^x \tag{2}$$

Performance Model:

In this work, we model the performance as inversely proportional to the number of skipped jobs. Furthermore, to formulate a minimization problem, performance maximization is formulated as the minimization of the penalty due to skipping jobs.

Let $\mathcal{S}_i | 0 \leq \mathcal{S}_i \leq n_i$ denotes the maximum number of jobs of T_i in H. We use a quadratic penalty function in terms of the number of misses. We define the quadratic penalty for T_i as $\mathcal{J}_i = (\mathcal{S}_i)^2$ and the total penalty is

$$\hat{\mathcal{J}} = \sum_i \mathcal{S}_i^2 = \sum_i \sum_x (c_i^o)^2 \cdot (1 - y_{ix}) \tag{3}$$

Objective: The objective is to optimize the energy consumption (Eq. 2) and performance (Eq. 3) jointly. Firstly, we use *min-max* normalization to transform E and $\mathcal{J}$ to the range [0,1]. Let, E^{min} and E^{max} denote the lower and upper bounds for E, respectively. $E^{min} = \sum_i E_i^{min}$ where E_i^{min} is computed assuming all mandatory portion of each job are executed with the smallest frequency. Thus, for the purpose of computing a lower bound, we compute E_i^{min} by assuming all mandatory portion of J_{ix} jobs in $\frac{H}{p_i}$ with smallest frequency $\min_j\{F_j^1\}$. For the sake of computing a theoretical bound, the timing constraint is ignored. Likewise, $E^{max} = \sum_i E_i^{max}$ where E_i^{max} is computed assuming that mandatory and optional jobs are accepted ($\mathcal{S}_i = 0$) and scheduled with the maximum frequency ($f_{ix} = 1$). Thus, normalized energy consumption as

$$\hat{E} = \frac{E - E^{min}}{E^{max} - E^{min}} \tag{4}$$

Likewise, $\mathcal{J}$ is normalized (to $\hat{\mathcal{J}}$) for which lower and upper bounds are computed assuming minimum misses $\mathcal{S}_i = 0$ and maximum misses $\mathcal{S}_i = n_i$ corresponding to each task, respectively.

B. Problem Statement

EOP problem is to schedule hard real-time tasks, to optimize performance and energy consumption jointly. However, scheduling hard real-time tasks is an intractable problem [**?**], [**?**]. EOP problem is more challenging as it involves optimizing the conflicting objectives, namely, energy consumption and performance. Furthermore, all jobs of a task are equally important, thus we assign job-level processor and frequency. Note this also allows us to utilize (and explore) a more extensive search space (compared to task-to-processor/frequency assignment). The focus of this work is to determine job-to-processor mapping, and job-to-frequency mapping.

Given a set of non-preemptive periodic hard real-time tasks $\{T_i\}$, and a set of DVFS-enabled heterogeneous multiprocessor system $\mathcal{P}$, the EPW problem is to (2) job to processor mapping, and (3) jobs to frequency mapping, to optimize the energy consumption and performance jointly (as defined in Eq. **??**) such that

- Timing constraint of mandatory and optional execution of jobs must be satisfied.
- Scheduling constraints are satisfied, specifically, (i) a processor may execute at most one job at a time in a non-preemptive manner, and (ii) each accepted job (mandatory and optional) is executed on exactly one processor

Final Multi-Objective Formulation:

$$minimize F = (f_1, f_2) \tag{5}$$

Constraints

a) 1. Timing Constraint::

$$\sum_j \sum_f (z_{ix}^{jf} \cdot \frac{c_i^m + y_{ix} \cdot c_i^o}{f} \leq p_i) \quad \forall i, x$$

b) 2. Unique Assignment Constraint::

$$\sum_j \sum_f z_{ix}^{jf} = 1 \quad \forall i, x$$

c) 4. Non-Preemptive Constraint:: For any two overlapping jobs J_{ix} and $J_{i'x'}$ assigned to the same processor P_j, ensure:

$$\text{Either } \text{end}_{ix} \leq \text{start}_{i'x'} \quad \text{or} \quad \text{end}_{i'x'} \leq \text{start}_{ix}$$

IV. NSGA-II

V. NSGA-II and Hybrid

VI. Heuristic

VII. Results

A. Input Generation

VIII. Conclusion

References