A Unified Framework for Period and Priority Optimization in Distributed Hard Real-Time Systems

Yecheng Zhao, Vinit Gala, Haibo Zeng

***Abstract*—Modern embedded systems such as automotive are physically distributed with an increasing number of microcontrollers and buses. They support complex functions such as active safety and autonomous driving features with a high degree of data dependencies. The most common configuration uses periodic activation of tasks and messages coupled with priority-based scheduling. Selecting task and message parameters so that end-to-end deadlines are met can be very challenging, since such deadlines are enforced across a set of microcontrollers and buses.**

In this paper, we address the problem of optimal selection of task and message activation periods and priorities. Existing approaches cannot scale to large designs and have to settle to optimize period or priority separately, largely due to the complexity of response time analysis techniques. Instead, we present a new, unified framework that simultaneously optimizes period and priority assignment. It avoids the pitfalls of existing approaches by abstracting the response time calculation with the new concept of Maximal Unschedulable Period and Deadline Assignment (MUPDA). We demonstrate with two industrial case studies that our approach runs magnitudes faster than existing approach on period optimization, while providing substantially better solutions.

I. Introduction

Modern embedded systems in application domains such as avionics, automotive, and smart buildings contain complex functional contents deployed on a distributed platform. For example, new active safety and autonomous driving features are integrated in today's vehicles, which collect data from 360° sensors (e.g., camera, radar, and LIDAR) to understand the position of surrounding objects and detect hazardous conditions. Once a hazard is detected, they inform the driver and/or provide control overlays to reduce the risk. Two examples are adaptive cruise control and lane keeping systems. These embedded systems have the following characteristics:

- Functional dependencies are modeled by a complex graph rather than a set of linear transactions. At the user level, timing constraints and performance metrics are expressed on end-to-end paths from sensors to actuators. In addition, sensor, control, and actuator functions operate with their own periodic tasks, with constraints on periods imposed by the need for stability and accuracy.
- The system is inherently multi-rate, because of technological constraints (e.g. off-the-shelf sensors operate at different rates), but also because the same inputs and outputs are shared by multiple control functions, characterized by different control laws with their period constraints. Merging flows with different rates and communication with oversampling/undersampling are common.

In this paper, we consider the optimization of period and priority assignment, a problem common in the system-level design stage of such embedded systems. Specifically, given an allocation of tasks and messages, our approach automatically assigns periods and priorities to all tasks and messages, in order to satisfy the period constraints and hard real-time constraints including end-to-end deadline requirements. Clearly, the solution quality depends on both and, ideally, the two decision variables should be optimized at once. However, in the past this optimization problem is considered to be too large, such that an integrated problem formulation cannot be solved in feasible time [7]. Thus, existing approaches are to optimize periods and priorities separately and possibly iteratively.

On the contrary, we develop a unified framework capable of co-optimizing both periods and priorities for large industrial designs. Our observation is that existing approaches try to directly leverage standard mathematical programming frameworks such as geometric programming (GP), but they face substantial difficulty due to the complexity of response time analysis. Instead, we establish an abstraction layer which hides the details of response time analysis but still faithfully respects its accuracy. This allows us to prudently combine the power of commercial integer linear programming (ILP) solver for generic branch-and-bound based search and customized algorithms to explore problem-specific optimization structures.

The *contributions and paper organization* are as follows. Section II discusses the related work. Section III presents the system model including a summary on the analysis of the timing metrics, and defines the optimization problem. Section IV introduces a set of concepts including Maximal Unschedulable Period and Deadline Assignment (MUPDA), to accurately capture the real-time schedulability conditions. Section V presents an optimization framework based on these concepts, to judiciously combine the efficient algorithm for calculating MUPDA and ILP solver for generic branch-and-bound. Section VI applies the framework to two industrial case studies. Compared to an existing GP-based approach that only optimizes the periods, our approach runs up to 100× faster while providing much better solutions. Finally, the paper is concluded in Section VII.

II. Related Work

The problem of priority assignment in hard real-time systems scheduled with fixed priority has been studied extensively.

This article was presented in the International Conference on Embedded Software 2018 and appears as part of the ESWEEK-TCAD special issue.

Yecheng Zhao, Vinit Gala, Haibo Zeng are with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, 24060, USA (E-mail: {zyecheng, vinitg, hbzeng}@vt.edu).

See an authoritative survey by Davis et al. [11]. Among them, Audsley's algorithm [1] is optimal for finding a schedulable priority assignment for a variety of task models and scheduling policies, as summarized in Davis et al. [11]. The three necessary and sufficient conditions for its optimality are presented in [9]. Besides schedulability, Audsley's algorithm can be revised to optimize several other objectives, including the number of priority levels [1], lexicographical distance (the perturbation needed to make the system schedulable from an initial priority order) [6], [11], robustness (ability to tolerate additional interferences) [8], and as a subproblem of this paper, the average worst case response time [33].

For complex problems on priority assignment optimization where Audsley's algorithm do not apply, the current approaches include (a) meta heuristics such as simulated annealing (e.g., [24], [3], [30]) and genetic algorithm (e.g., [16], [27]); (b) problem specific heuristics (e.g., [22], [29], [25]); and (c) directly applying existing optimization frameworks such as branch-and-bound (BnB) (e.g., [26]) and ILP (e.g., [14], [31], [28]). These approaches either have no guarantee on solution quality, or suffer from scalability issues and may have difficulty to handle large industrial designs.

Such approaches are also followed for the problem of period optimization or the co-optimization of period and priority. On *single-core platforms*, examples include [23], [5], [4], [18]. [23] approximates the schedulability condition with a utilization bound, hence the solution may be arbitrarily suboptimal. Bini et al. [5] develop a branch-and-bound (BnB) based algorithm and a fast suboptimal heuristic. In another work, Bini et al. [4] derive analytical solutions that are specific for period assignment to optimize a particular form of control performance, and the method relies on an approximate response time analysis. A BnB algorithm is built on top of [4] to additionally find the best priority assignment [18]. Davare et al. [7] consider a simpler problem than this paper, the period optimization in *distributed* hard real-time systems. They formulate it in mixed integer GP (MIGP) framework, and also propose an iterative procedure that relies on an approximate, direct formulation in GP. This procedure is also leveraged in several recent works on period optimization, such as [12].

Like our approach, Zhao et al. develop customized optimization procedures that are exact and efficient [32], [33], [34]. However, these works [32], [33], [34] consider the problem of priority assignment and assume periods are fixed. *Different from all the above*, we are the first to simultaneously optimize periods and priorities in distributed hard real-time systems with end-to-end deadline constraints. Although we also build a customized procedure, the concepts and algorithms from [32], [33], [34] are not directly applicable.

III. System Model and Notation

We consider a distributed real-time system represented by a directed acyclic graph $\Gamma = \{\mathcal{V}, \mathcal{L}\}$. $\mathcal{V} = \{\tau_1, \cdots, \tau_n\}$ denotes the set of objects, each representing a scheduling entity (i.e., task or message). $\mathcal{L}$ is the set of directed edges representing the communication flow between the objects. Also, $\mathcal{R} = \{r_1, \cdots, r_c\}$ is the set of (possibly heterogeneous) resources supporting the execution of the objects, i.e., r_i is a microcontroller for task execution or a bus for message transmission. In this work, we assume that mapping of objects to resources is fixed, and is not part of the decision variables.

An object τ_i is characterized by a worst-case execution time (WCET) C_i, an activation period T_i, a deadline D_i, and a scheduling priority π_i. We assume that C_i, D_i and T_i take only integer values. We do not assume any particular type of resources as long as they are scheduled with partitioned fixed priority, and the objects can be either *preemptive* or *non-preemptive*. Such scheduling policies are widely adopted in different real-time applications and standards, such as automotive AUTOSAR/OSEK real-time operating systems (RTOS) standard, the modern RTOSes including LynxOS and QNX, and the Controller Area Network (CAN) protocol and its extension CAN-FD (CAN with Flexible Data-rate). τ_i is *schedulable* if its worst-case response time (WCRT) or simply response time, denoted as R_i, is no larger than its deadline D_i.

A directed edge $\langle \tau_i, \tau_j \rangle$ exists in $\mathcal{L}$ if τ_i writes to τ_j. At the start of execution, τ_j samples the input of τ_i, which are then processed during its execution. Upon completion, the result is delivered to its output for its successors to sample. An end-to-end path p in the system consists of a set of directed edges connecting from a source to a sink, which represents a chain of communicating objects. One semantics of the end-to-end latency of path p is the total amount of time from the instant when the input data is first sampled by the source object to the instant when the output is produced by the sink object. For periodic activation, the worst case end-to-end latency is the summation of the processing latency of each object τ_j, i.e., $\sum_j l_j$ where $l_j = R_j + T_j$ [7].

Intuitively, this can be understood as a scenario where τ_j just misses the output of τ_i produced at the start of its current execution and thus has to wait until the next activation to sample it. This introduces a sampling latency equal to the period of τ_j. Correspondingly, the end-to-end latency of path p_i is then [7]

$$L_p = \sum_{\forall j \in p} l_i = \sum_{\forall j \in p} (R_i + T_i) \tag{1}$$

For multiple communicating tasks with harmonic periods on the same microcontroller, the analysis can be less pessimistic if the designer can select the relative activation phase of all tasks [13]. In addition, other semantics on end-to-end latency may exist, and we refer the readers to [15].

Each path p is characterized by an end-to-end deadline D_p requiring that $L_p \leq D_p$. A correct design of the system should satisfy not only the schedulability of each object, but also the end-to-end deadline constraints for all paths.

We now provide a summary on the response time analysis. For fixed priority preemptive scheduling with constrained (i.e., $D_i \leq T_i$) or implicit (i.e., $D_i = T_i$) deadline, the WCRT R_i of an object τ_i is the least fixed-point of the following equation

$$R_i = C_i + \sum_{\forall \tau_j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{2}$$

where $hp(i)$ represents the set of higher priority objects allocated on the same execution platform as τ_i.

For arbitrary deadline, R_i may not occur in the first instance in the busy period, and it is necessary to check all instances in the busy period. Specifically, R_i is computed as follows

$$\begin{aligned} R_i(q) &= (q+1)C_i + \sum_{\forall \tau_j \in hp(i)} \left\lceil \frac{R_i(q)}{T_i} \right\rceil C_j \\ R_i &= \max_q \{R_i(q) - qT_i\} \\ &\text{where } q = 0...q^* \text{ until } R_i(q^*) \leq (q^*+1)T_i \end{aligned} \tag{3}$$

For non-preemptive scheduling, we summarize the accurate analysis and a safe approximation proposed in [10]. Specifically, it is necessary to check all instances in the busy period even if the object has constrained deadline. τ_i now suffers a worst case blocking time equal to the maximum WCET from the lower priority objects

$$B_i = \max_{\forall \tau_j \in lp(i)} \{C_j\} \tag{4}$$

where $lp(i)$ is the set of lower priority objects allocated on the same platform as τ_i. The longest busy period t_i^b at the priority level of τ_i is the fixed point of the following equation

$$t_i^b = B_i + \left\lceil \frac{t_i^b}{T_i} \right\rceil C_i + \sum_{\forall \tau_j \in hp(i)} \left\lceil \frac{t_i^b}{T_j} \right\rceil C_j \tag{5}$$

The WCRT of τ_i is then computed as follows

$$\begin{aligned} w_i(q) &= qC_i + B_i + \sum_{\forall \tau_j \in hp(i)} \left\lceil \frac{w_i(q)}{T_j} \right\rceil C_j \\ R_i &= \max_{q=0...q^*} \{w_i(q) - qT_i + C_i\} \text{ where } q^* = \left\lceil \frac{t_i^b}{T_i} \right\rceil - 1 \end{aligned} \tag{6}$$

For constrained deadline, τ_i can either suffer the blocking of a lower priority object or the push through interference from the previous instance of the same object, but not both [10]. Hence, it is sufficient to only check the first instance in the busy period

$$R_i = C_i + \hat{B}_i + \sum_{\forall \tau_j \in hp(i)} \left\lceil \frac{R_i - C_i}{T_j} \right\rceil C_j \tag{7}$$

where $\hat{B}_i$ is defined as $\hat{B}_i = \max\{C_i, B_i\}$.

A. Problem Definition

In this paper, we consider the design optimization problem where the decision variables include the set of periods $\mathbf{T}$ and priority assignments $\mathbf{P}$ for all tasks/messages. The feasibility constraints include the schedulability of each task/message and the end-to-end deadline requirements for all critical paths. Moreover, the period assignments must maintain harmonicity for the specified pairs of objects. This can be enforced by the following constraint

$$T_i = h_{i,j} \cdot T_j, \quad \forall \text{ harmonicity pair } \tau_i, \tau_j \tag{8}$$

where $h_{i,j}$ represents the harmonicity factor and is integral. When $h_{i,j}$ is a given constant, (8) is simply a linear constraint. When $h_{i,j}$ is also a decision variable (i.e,. the designer is allowed to choose the harmonicity factor), (8) becomes a quadratic integer constraint. Such harmonicity constraints may be motivated by the possibility to reduce the end-to-end latency [13], but also may be imposed by development tools such as Simulink (which requires any pair of communicating functions have harmonic periods [19]).

Also, period bounds may be specified, especially for feedback control applications

$$T_i^{\text{lb}} \leq T_i \leq T_i^{\text{ub}}, \ \forall \tau_i \tag{9}$$

Finally, the total utilization of an execution platform r_k may not exceed a specified threshold $U_k^{\max}$ for future extensibility.

$$\sum_{\tau_i \in r_k} \frac{C_i}{T_i} \leq U_k^{\max}, \ \forall r_k \tag{10}$$

Formally, the problem can be expressed as follows.

$$\begin{aligned} &\min_{\forall \mathbf{X}} F(\mathbf{X}) \\ &s.t. \text{ Schedulability: } R_i \leq D_i, \ \forall \tau_i \\ &L_p \leq D_p, \ \forall p \\ &(8) - (10) \end{aligned} \tag{11}$$

where $\mathbf{X}$ represents the set of decision variables including the periods $\mathbf{T}$ and priorities $\mathbf{P}$ of the tasks and messages. $F(\mathbf{X})$ represents an optional objective function. In this paper, we consider the objective of minimizing the average WCRT over a selected set of tasks/messages Ω, as adopted in several previous works [7], [33]

$$F(\mathbf{X}) = \sum_{\forall \tau_i \in \Omega} R_i \tag{12}$$

This metric is a quantification of the responsiveness of the selected tasks/messages. Although our framework may be extended to other objectives, we leave it to future work.

IV. The Concept of MUPDA

Before presenting our approach, we first discuss the challenges and the possible drawbacks from existing exact algorithms on period optimization for distributed hard real-time systems [7]. Specifically, [7] (like many other works on design optimization of hard real-time systems, as discussed in Section II) tries to leverage standard mathematical programming framework, including a direct formulation of the response time analysis. This suffers from the following issues.

- Response time analysis is notoriously inefficient to formulate in standard mathematical programming framework. Consider the analysis in (2) and the simplified problem of optimizing period (as addressed in [7]). It has been shown that a mixed integer geometric programming (MIGP) formulation of the analysis requires $O(n^2)$ number of integer variables, each for calculating the number of interferences $\left\lceil \frac{R_i}{T_j} \right\rceil$ [7]. This makes the formulation difficult to solve even for small and medium size problems. To avoid this difficulty, [7] introduces

TABLE I: An Example Task System Γ_e

τ_i	T_i^{lb}	T_i^{ub}	C_i	τ_i	T_i^{lb}	T_i^{ub}	C_i
τ_1	0	10	2	τ_2	0	20	3
τ_3	0	40	10	τ_4	0	100	3

an iterative procedure but still relies on a direct formulation in GP. Specifically, it introduces an additional set of parameters $\alpha_{i,j}$ and approximates the WCRT analysis as follows

$$R_i' = C_i + \sum_{\forall \tau_j \in hp(i)} (\frac{R_i'}{T_j} + \alpha_{i,j}) C_j \tag{13}$$

The analysis is then formulated as a geometric program integrated into an iterative procedure to adjust $\alpha_{i,j}$. The cost of these approximations however, is a possible loss of optimality and sometimes the procedure may not even converge.

• The applicability of these approaches is typically limited to the schedulability analysis such as (2) and (7), which only needs to evaluate the first instance in the busy period for estimating WCRT. For more sophisticated scenario (e.g., arbitrary deadline setting) that requires analysis like (3) and (6), the major difficulty is that the number of instances q^* in the busy period cannot be determined in advance, as the length of the busy period is unknown a priori when periods are variables. This essentially makes it impractical for any possible formulation in standard mathematical programming.

• When priority assignment is also part of the decision variables, the problem becomes even more challenging. MIGP is no longer able to handle it due to the additional constraints from priority assignment. For example, the asymmetric constraints require that if τ_i has a higher priority than τ_j ($p_{i,j} = 1$), then τ_j must have a lower priority than τ_i ($p_{j,i} = 0$) [32]. Such constraints have the form of $p_{i,j} + p_{j,i} = 1$, which is incompatible with MIGP [20]. This hinders the possibility of achieving significant improvement of optimization quality brought by co-optimizing both period and priority assignment.

Instead, we propose a technique that avoids the above pitfalls in existing approaches. The main idea is to use a set of compact constraints for schedulability that hides the details of the underlying schedulability analysis from the mathematical programming solver. Our approach is applicable to two scenarios. The *first* assumes that priority assignment is given and periods are the decision variables. The *second* considers the more general problem where both periods and priority assignments are decision variables. Both variants of the problem can be solved by the proposed framework. For the first, the proposed technique is optimal w.r.t. the objective function for any schedulability analysis that is sustainable w.r.t. periods and deadlines (e.g., all the analyses summarized in Section III). For the second version, the proposed technique preserves optimality with the WCRT analysis in (2) and (7), and is close to optimal for others.

In this section, we introduce a set of concepts for abstracting the schedulability conditions, including Maximal Unschedulable Period-Deadline Assignment (MUPDA). MUPDA is an extension of the concept of MUDA (maximal unschedulable deadline assignment) proposed in [33], by adding period assignment information.

For clarity, we use an illustrative example system in Table I in this section and Section V. All the four tasks are allocated on the same execution platform. They are preemptive with implicit deadline, hence the WCRT analysis in (2) is accurate. There is one end-to-end path $\tau_2 \rightarrow \tau_3$ with a deadline requirement of 63. There is no harmonicity constraint or utilization bound. Both periods $\mathbf{T}$ and priorities $\mathbf{P}$ are decision variables, thus the design optimization of the example system is

$$\begin{aligned} &\min_{\forall \mathbf{T},\mathbf{P}} R_1 + R_2 + R_3 + R_4 \\ s.t.\ &\text{Schedulability} : R_i \le T_i, \forall \tau_i \\ &R_2 + T_2 + R_3 + T_3 \le 63 \\ &T_i^{\text{lb}} \le T_i \le T_i^{\text{ub}},\ \forall \tau_i \end{aligned} \tag{14}$$

Definition 1. [33] A *Virtual Deadline (VD)* is a tuple $\langle \tau_i, d_i \rangle^D$ where d_i is a positive integer, which represents an over-estimated WCRT $R_i \le d_i$. A *WCRT summation bound* is a tuple $\langle \Omega, d \rangle^W$ where d is a positive integer. It represents the following constraint

$$\sum_{\forall \tau_i \in \Omega} R_i \le d \tag{15}$$

Intuitively, in $\langle \tau_i, d_i \rangle^D$, d_i is an estimated value on the WCRT R_i that shall be pessimistic (such that we will not give false positive on the schedulability of τ_i). Similarly, d in $\langle \Omega, d \rangle^W$ is an over-estimation on the objective (the summation of WCRTs).

Definition 2. A *period assignment* is a tuple $\langle \tau_i, t_i \rangle^T$ where $t_i \le T_i^{\text{ub}}$ is a positive integer. It represents that the period of τ_i is assigned to be t_i, namely $T_i = t_i$.

Definition 3. A *period-deadline assignment*, or shortly a T-D assignment $\mathcal{R}$ is a collection of (i) a virtual deadline $\langle \tau_i, d_i \rangle^D$ for each τ_i, (ii) a period assignment $\langle \tau_i, t_i \rangle^T$ for each τ_i, and (iii) a WCRT summation bound $\langle \Omega, d_\Omega \rangle^W$. Namely, $\mathcal{R}$ can be expressed as

$$\mathcal{R} = \{ \langle \tau_1, d_1 \rangle^D .. \langle \tau_n, d_n \rangle^D , \langle \tau_1, t_1 \rangle^T .. \langle \tau_n, t_n \rangle^T , \langle \Omega, d_\Omega \rangle^W \}$$

The following definition gives a *partial* order relationship among period-deadline assignments.

Definition 4. $\mathcal{R}_1$ is said to *dominate* $\mathcal{R}_2$, denoted as $\mathcal{R}_1 \succeq \mathcal{R}_2$, if the following conditions hold.

$$\begin{aligned} &d_i \ge d_i',\ \forall \langle \tau_i, d_i \rangle^D \in \mathcal{R}_1, \langle \tau_i, d_i' \rangle^D \in \mathcal{R}_2 \\ &t_i \ge t_i',\ \forall \langle \tau_i, t_i \rangle^T \in \mathcal{R}_1, \langle \tau_i, t_i' \rangle^T \in \mathcal{R}_2 \\ &d_\Omega \ge d_\Omega', \langle \Omega, d_\Omega \rangle^W \in \mathcal{R}_1, \langle \Omega, d_\Omega' \rangle^W \in \mathcal{R}_2 \end{aligned} \tag{16}$$

Equivalently, $\mathcal{R}_1 \succeq \mathcal{R}_2$, if and only if $\mathcal{R}_1$ is component-wise no smaller than $\mathcal{R}_2$. $\mathcal{R}_1$ is said to *strictly dominate* $\mathcal{R}_2$, denoted as $\mathcal{R}_1 \succ \mathcal{R}_2$, if $\mathcal{R}_1 \succeq \mathcal{R}_2$ and $\mathcal{R}_1 \neq \mathcal{R}_2$.

Example 1. Consider the following T-D assignments for Γ_e

$$\begin{aligned}\mathcal{R}_1 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,20\rangle^D,\langle\tau_3,40\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,40\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,170\rangle^W\}\\ \mathcal{R}_2 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,15\rangle^D,\langle\tau_3,40\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,40\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,170\rangle^W\}\\ \mathcal{R}_3 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,20\rangle^D,\langle\tau_3,30\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,40\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,170\rangle^W\}\end{aligned}$$

$\mathcal{R}_1 \succeq \mathcal{R}_2$ and $\mathcal{R}_1 \succeq \mathcal{R}_3$, as the virtual deadlines, period assignments and WCRT summation bound in $\mathcal{R}_1$ is component wise no smaller than those of $\mathcal{R}_2$ and $\mathcal{R}_3$. Also, neither $\mathcal{R}_2 \succeq \mathcal{R}_3$ nor $\mathcal{R}_3 \succeq \mathcal{R}_2$: compared to $\mathcal{R}_2$, $\mathcal{R}_3$ has a larger virtual deadline on τ_2 but a smaller virtual deadline on τ_3. Thus, Definition 4 defines a partial order among all T-D assignments.

Definition 5. Let $\mathcal{R} = \{\langle\tau_1,d_1\rangle^D,...\langle\tau_n,d_n\rangle^D, \langle\tau_1,t_1\rangle^T,...\langle\tau_n,t_n\rangle^T, \langle\Omega,d_\Omega\rangle^W\}$ be a T-D assignment. The system Γ is *$\mathcal{R}$-schedulable*, or informally $\mathcal{R}$ is *schedulable*, if and only if there exists a priority assignment such that (i) $T_i = t_i, \forall \langle\tau_i,t_i\rangle^T \in \mathcal{R}$; (ii) $R_i \le d_i, \forall \langle\tau_i,d_i\rangle^D \in \mathcal{R}$; and (iii) $\sum_{\tau_i\in\Omega} R_i \le d_\Omega$. That is, $\mathcal{R}$ is schedulable if and only if there exists a priority assignment that respects all the assignments on periods, virtual deadlines, and WCRT summation bound in $\mathcal{R}$.

Example 2. Consider the following T-D assignments

$$\begin{aligned}\mathcal{R}_1 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,20\rangle^D,\langle\tau_3,40\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,40\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,170\rangle^W\}\\ \mathcal{R}_2 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,15\rangle^D,\langle\tau_3,2\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,40\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,170\rangle^W\}\\ \mathcal{R}_3 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,15\rangle^D,\langle\tau_3,2\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,40\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,18\rangle^W\}\end{aligned}$$

$\mathcal{R}_1$ assigns the most relaxed period, virtual deadlines and WCRT summation bound and is schedulable by rate-monotonic priority assignment. $\mathcal{R}_2$ is obviously unschedulable as the virtual deadline on τ_3 cannot even accommodate its worst-case execution time. $\mathcal{R}_3$ differs from $\mathcal{R}_1$ in the WCRT summation bound. Though individual virtual deadlines can be satisfied for $\mathcal{R}_3$ by rate-monotonic priority assignment, its WCRT summation bound, which equals the summation of WCETs of all objects, obviously cannot be satisfied for any priority assignment. Thus $\mathcal{R}_3$ is also unschedulable.

We now reformulate the original problem (11) into the following form with the concept of $\mathcal{R}$-schedulability.

$$\begin{aligned}&\min_{\forall\mathcal{R}} d_\Omega\\ s.t.\ &\Gamma \text{ is } \mathcal{R}\text{-schedulable}\\ &L'_p \le D_p,\ \forall p\\ &(8)'-(10)'\end{aligned} \tag{17}$$

L'_p is calculated in the same way as L_p, but instead R_i is replaced with d_i where $\langle\tau_i,d_i\rangle^D \in \mathcal{R}$ and T_i is replaced with t_i where $\langle\tau_i,t_i\rangle^T \in \mathcal{R}$. Similarly, (8)'-(10)' are derived from (8)-(10) by replacing T_i with t_i.

Informally, the reformulated problem (17) is to find a schedulable $\mathcal{R}$ with minimum value on d_Ω that satisfies all the end-to-end latency deadline constraints and (8)-(10). The equivalence between (17) and (11) is straightforward. Consider a feasible solution of (11). Construct a period-deadline assignment $\mathcal{R}$ by setting $d_i = R_i$ for all virtual deadlines $\langle\tau_i,d_i\rangle^D$, $t_i = T_i$ for all period assignments $\langle\tau_i,t_i\rangle^T$ and $d_\Omega = \sum_{\forall\tau_i\in\Omega} R_i$. $\mathcal{R}$ is also a feasible solution of problem (17). Similarly, consider a feasible solution $\mathcal{R}$ of problem (17). Since $\mathcal{R}$-schedulability guarantees that Γ is feasible with $R_i \le d_i$ for all objects and $\sum_{\tau_i\in\mathcal{R}} R_i \le d_\Omega$ under period assignment $T_i = t_i$ for all τ_i, $\mathcal{R}$ also implies the existence of a feasible solution for (11).

We now introduce an abstraction scheme that efficiently models the feasibility region of $\mathcal{R}$-schedulability for (17).

Theorem 1. Let $\mathcal{R}$ be an unschedulable period-deadline assignment. Any $\mathcal{R}'$ such that $\mathcal{R} \succeq \mathcal{R}'$ is also unschedulable.

Proof. The schedulability analyses in (2)–(7) are all *sustainable* w.r.t. deadlines and periods of the objects [2], i.e., increasing the deadline or period of any object can only make the system more schedulable. The sustainability property also trivially extends to the WCRT summation bound d_Ω in $\mathcal{R}$, hence the proof. □

Theorem 1 implies that each unschedulable period-deadline assignment $\mathcal{R}$ captures also the unschedulability of other period-deadline assignments $\mathcal{R}'$ dominated by it. The usefulness of the theorem is that it generalizes from one unschedulable period-deadline assignment to a potentially large set of unschedulable ones. The following definition introduces a special type of period-deadline assignment that is "most general" in capturing unschedulability.

Definition 6. $\mathcal{U}$ is a *maximal unschedulable period-deadline assignment (MUPDA)* if it satisfies the following condition

- Γ is not $\mathcal{U}$-schedulable;
- For all $\mathcal{R}$ such that $\mathcal{R} \succ \mathcal{U}$, Γ is $\mathcal{R}$-schedulable.

Example 3. Consider the following T-D assignments

$$\begin{aligned}\mathcal{R}_1 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,20\rangle^D,\langle\tau_3,10\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,10\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,170\rangle^W\}\\ \mathcal{R}_2 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,20\rangle^D,\langle\tau_3,40\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,16\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,170\rangle^W\}\end{aligned}$$

$\mathcal{R}_1$ is unschedulable for the following reason. τ_1 must have higher priority than τ_3 as $C_3 \ge T_1^{\text{ub}}$. Under this constraint, the virtual deadline and period assignment for τ_3 cannot accommodate its schedulability. $\mathcal{R}_2$ is also unschedulable. Consider the rate-monotonic priority assignment, known to be optimal for schedulability of individual tasks. τ_2 suffers the interference of at least two instances of τ_1 and one instances of τ_3. Thus R_2 is at least 17. When $T_3 = 16$, τ_2 suffers one more instance of interference from τ_3, which violates its schedulability. However, If T_3 is increased by one (i.e., $T_3 = 17$), τ_2 will be schedulable. With the above result, $\mathcal{R}_1$

Algorithm 1 Algorithm for Computing MUPDA

1: **function** MUPDA(System Γ, Unschedulable $\mathcal{R}$)
2: **for** each element $\zeta \in \mathcal{R}$ (ζ may be $\langle \tau_i, v\rangle^D$ or $\langle \tau_i, v\rangle^T$ or $\langle \Omega, v\rangle^W$) **do**
3: Use binary search to find out the largest value u that v can be increased to while keeping $\mathcal{R}$ unschedulable
4: Update the value of v to be u
5: **end for**
6: return $\mathcal{R}$
7: **end function**

is not a MUPDA since $\mathcal{R}_2 \succeq \mathcal{R}_1$ and $\mathcal{R}_2$ is unschedulable. $\mathcal{R}_2$ is a MUPDA however, as all $\mathcal{R} \succeq \mathcal{R}_2$ (which only consist of $\mathcal{R}$s with larger period assignment on τ_3, as all other virtual deadlines, periods assignment and WCRT summation bound are at their upper bounds) are schedulable.

Remark 1. A MUPDA is a maximal generalization of unschedulable period-deadline assignment in the sense that there is no other unschedulable $\mathcal{R}$ that strictly dominates $\mathcal{U}$. MUPDAs are *not unique*: it is possible that a system Γ has multiple MUPDAs by Definition 6.

We now show how MUPDAs can be used to derive an abstract form of schedulability constraint for use in problem (17). A MUPDA $\mathcal{U} = \{\langle \tau_1, d'_1\rangle^D, ... \langle \tau_n, d'_n\rangle^D, \langle \tau_1, t'_1\rangle^T, ... \langle \tau_n, t'_n\rangle^T, \langle \Omega, d'_\Omega\rangle^W\}$ suggests that all period-deadline assignments

$$\mathcal{R} = \{\langle \tau_1, d_1\rangle^D .. \langle \tau_n, d_n\rangle^D, \langle \tau_1, t_1\rangle^T .. \langle \tau_n, t_n\rangle^T, \langle \Omega, d_\Omega\rangle^W\}$$

satisfying the following constraints are unschedulable

$$\begin{cases} d_i \le d'_i, & \forall \tau_i \\ t_i \le t'_i, & \forall \tau_i \\ d_\Omega \le d'_\Omega, \end{cases} \tag{18}$$

Contra-positively, the following constraints are necessary to be satisfied for any *schedulable* period-deadline assignment $\mathcal{R}$.

$$\mathcal{R} \not\preceq \mathcal{U} \quad \Leftrightarrow \quad \neg \begin{cases} d_i \le d'_i, \ \forall \tau_i \\ t_i \le t'_i, \ \forall \tau_i \\ d_\Omega \le d'_\Omega, \end{cases} \quad \Leftrightarrow \quad \begin{Vmatrix} d_i > d'_i, \ \forall \tau_i \\ t_i > t'_i, \ \forall \tau_i \\ d_\Omega > d'_\Omega, \end{Vmatrix} \tag{19}$$

where $\|$ represents the logical OR (disjunction) operation.

We call (19) MUPDA implied constraints by $\mathcal{U}$. Our general idea is to use (19) as the form of constraints for shaping the feasibility region of $\mathcal{R}$-schedulability in problem (17). The disjunction can be formulated as integer linear constraint [32].

We now discuss how MUPDAs can be obtained given an unschedulable $\mathcal{R}$. This is detailed in Algorithm 1. Specifically, Algorithm 1 is based on the property that the schedulability analysis is sustainable w.r.t. deadline, period, and WCRT summation. When performing the binary search for determining the largest value u (Line 2-3) on a particular element (virtual deadline, period assignment, or WCRT summation bound) in $\mathcal{R}$, the other elements are kept unchanged. If the system is still unschedulable even by setting v to the upper bound, then v is updated to the upper bound. The order of visit in Line 2 may affect the returned MUPDA in the sense that different orders may return different MUPDAs. To compute multiple MUPDAs from a single $\mathcal{R}$, it suffices to perturb $\mathcal{R}$ into $\mathcal{R}'$ such that any previously computed MUPDA $\mathcal{U}$ does not dominate $\mathcal{R}'$, i.e., $\mathcal{U} \not\succeq \mathcal{R}'$. This guarantees that the MUPDA computed from $\mathcal{R}'$ is different from all previous ones.

Example 4. We now illustrate Algorithm 1 by applying it to $\mathcal{R}_1$ in Example 3. Suppose the algorithm iterates through each element in $\mathcal{R}_1$ according to the order shown in the example. $\langle \tau_1, 10\rangle^D$ and $\langle \tau_2, 20\rangle^D$ are explored in the first two iterations, but they already at their upper bound, thus nothing is performed. In the third iteration, the algorithm considers $\langle \tau_3, v\rangle^D$ where $v = 10$. It uses binary search to find out the maximum value v can be increased to while maintaining unschedulability, assuming all other elements in $\mathcal{R}_1$ remain unchanged. By the reasoning in Example 3, even if v is increased to the upper bound 40, τ_2 is still unschedulable due to $\langle \tau_3, 10\rangle^T$. Thus $\langle \tau_3, 10\rangle^D$ is updated to $\langle \tau_3, 40\rangle^D$. Similarly, when the algorithm iterates on $\langle \tau_3, 10\rangle^T$, it discovers that the system becomes schedulable after T_3 is increased to 17. Thus it updates $\langle \tau_3, 10\rangle^T$ to $\langle \tau_3, 16\rangle^T$, the largest value that maintains unschedulability. In the end, given $\mathcal{R}_1$ as input, Algorithm 1 finds the MUPDA $\mathcal{R}_2$ shown in Example 3.

Next we consider computing a second MUPDA from $\mathcal{R}_1$. The key is to perturb $\mathcal{R}_1$ into $\mathcal{R}'_1$ such that $\mathcal{R}'_1 \not\preceq \mathcal{R}_2$. This can be done, for example, by setting $\langle \tau_3, 10\rangle^T$ in $\mathcal{R}_1$ to $\langle \tau_3, 17\rangle^T$, which gives

$$\begin{aligned} \mathcal{R}'_1 = \{&\langle \tau_1, 10\rangle^D, \langle \tau_2, 20\rangle^D, \langle \tau_3, 10\rangle^D, \langle \tau_4, 100\rangle^D \\ &\langle \tau_1, 10\rangle^T, \langle \tau_2, 20\rangle^T, \langle \tau_3, 17\rangle^T, \langle \tau_4, 100\rangle^T, \langle \Omega, 170\rangle^W\} \end{aligned}$$

$\mathcal{R}'_1$ is still unschedulable as the deadline assignment $\langle \tau_3, 10\rangle^D$, which equals C_3, is too small. Applying Algorithm 1 gives the following MUPDA.

$$\begin{aligned} \mathcal{R}'_2 = \{&\langle \tau_1, 10\rangle^D, \langle \tau_2, 20\rangle^D, \langle \tau_3, 11\rangle^D, \langle \tau_4, 100\rangle^D \\ &\langle \tau_1, 10\rangle^T, \langle \tau_2, 20\rangle^T, \langle \tau_3, 17\rangle^T, \langle \tau_4, 100\rangle^T, \langle \Omega, 170\rangle^W\} \end{aligned}$$

Algorithm 1 requires an efficient procedure to check the schedulability of $\mathcal{R}$ (Line 3). As mentioned earlier in this section, we consider two scenarios. The first assumes that priority assignment is given. In this case, $\mathcal{R}$-schedulability is straightforward to test, by (i) setting the period of each object according to the period assignment in $\mathcal{R}$; (ii) computing the WCRT R_i of each object τ_i as well as the summation $\sum_{\forall \tau_i \in \Omega} R_i$; and (iii) verifying if all constraints on schedulability and WCRT summation are satisfied by $\mathcal{R}$. The procedure is summarized in Algorithm 2. In this scenario, the algorithm is exact w.r.t. to any given response time analysis.

The second scenario assumes the priority assignments are also variables. This is harder as an exact $\mathcal{R}$-schedulability test requires to solve an optimization problem as below

$$\begin{aligned} &\min_{\forall \mathbf{P}} \sum_{\forall \tau_i \in \Omega} R_i \\ &s.t. \ \text{ System } \Gamma \text{ is schedulable} \end{aligned} \tag{20}$$

Algorithm 2 $\mathcal{R}$-schedulability test with priority assignment

1: **function** $\mathcal{R}-$SCHEDULABILITY(System Γ, T-D Assignment $\mathcal{R}$)
2: Set $T_i = t_i$ for all $\langle \tau_i, t_i \rangle^T \in \mathcal{R}$
3: Compute WCRT R_i for each object τ_i
4: **if** $R_i \leq d_i, \forall \langle \tau_i, d_i \rangle^D \in \mathcal{R}$ **then**
5: **if** $\sum_{\forall \tau_i \in \Omega} R_i \leq d_\Omega$, where $\langle \Omega, d_\Omega \rangle^W \in \mathcal{R}$ **then**
6: **return** true
7: **end if**
8: **end if**
9: **return** false
10: **end function**

Algorithm 3 $\mathcal{R}$-schedulability test without priority assignment

1: **function** $\mathcal{R}-$SCHEDULABILITY(System Γ, T-D Assignment $\mathcal{R}$)
2: Set $T_i = t_i$ for all $\langle \tau_i, t_i \rangle^T \in \mathcal{R}$.
3: Set $D_i = d_i$ for all $\langle \tau_i, d_i \rangle^D \in \mathcal{R}$.
4: Assign priorities according to [33, Algorithm 1]
5: Compute WCRT R_i for each object τ_i
6: **if** $R_i \leq d_i, \forall \langle \tau_i, d_i \rangle^D \in \mathcal{R}$ **then**
7: **if** $\sum_{\forall \tau_i \in \Omega} R_i \leq d_\Omega$, where $\langle \Omega, d_\Omega \rangle^W \in \mathcal{R}$ **then**
8: **return** true
9: **end if**
10: **end if**
11: **return** false
12: **end function**

The optimal objective is then compared to the WCRT summation bound d_Ω specified in $\mathcal{R}$, which determines whether $\mathcal{R}$ is schedulable. Though the problem is generally difficult, [33] shows that for systems with constrained deadlines using response time analyses in (2) and (7), a variant of Audsley's algorithm that always consider tasks with larger WCET first at each priority level is optimal for the above problem. For arbitrary deadline setting or response time analyses in (3) and (6), this algorithm does not guarantee optimality but is typically very close to optimal[33].

Algorithm 3 summarizes our proposed procedure for testing $\mathcal{R}$-schedulability in the second scenario. It differs from Algorithm 2 in Lines 3–4: in Line 3 Algorithm 3 updates the deadline of each object to be the virtual deadline in $\mathcal{R}$, which is necessary for performing priority assignment in Line 4.

Another issue is that the total number of MUPDAs for a system may be exponential to the number of tasks. It is obviously impractical to compute all of them and add the implied constraint to problem (17). However, we observe that in most cases, not all MUPDAs are related to the objective and end-to-end latency constraints. In fact, the optimal solution of (17) can usually be defined by a small number of MUPDA implied constraints. In the next section, we propose an iterative procedure that prudently explores only a subset of all MUPDAs that are sufficient to establish the optimal solution.

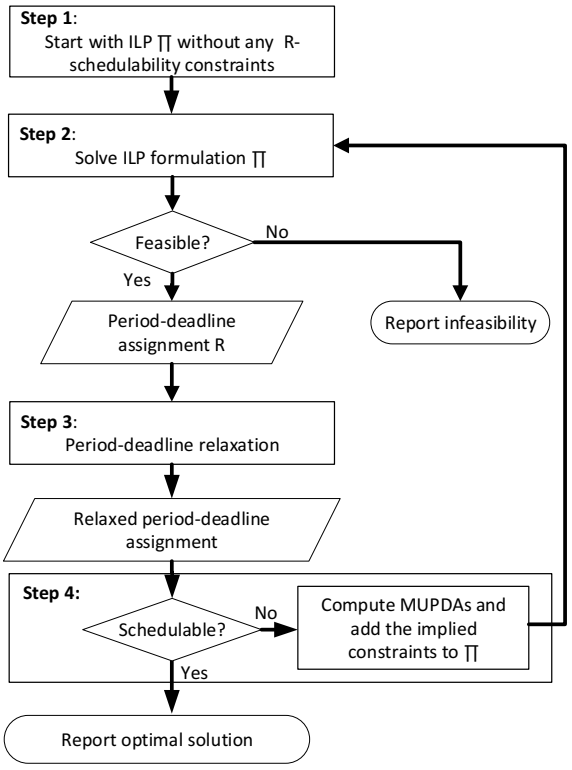


Fig. 1: The MUPDA guided optimization framework.

V. MUPDA Guided Optimization

We now present the complete optimization framework, an iterative procedure as summarized in Figure 1. The basic idea is to leverage ILP solvers for generic branch-and-bound based search and the efficient algorithms for calculating MUPDAs. Specifically, at any point of the procedure execution where a subset $\mathbb{U}$ of all MUPDAs are calculated, we partition the problem into two parts: (i) a relaxation Π of the problem (17) that includes MUPDA implied constraints from $\mathbb{U}$ (and implicitly part of the schedulability conditions), the end-to-end deadline constraints, and (8)'–(9)', which will be handled by the ILP solver; and (ii) those constraints not included in Π, which will be handled by the algorithms in Section IV.

To make Π compatible with ILP, the MUPDA implied constraints, as in the form of (19), can be formulated as integer linear constraints by adding a set of auxiliary binary variables [32]. Also, linearization techniques [21] can convert (8)' to integer linear constraints.

Finally, we enforce (10)' by modifying Algorithms 2 and 3. Specifically, before verifying schedulability, the total utilization of each execution platform is checked. If any utilization bound is violated, the system is considered to be $\mathcal{R}$-unschedulable. In this sense, we extend the definition of $\mathcal{R}$-schedulability to include also the utilization bound constraints in addition to the system schedulability. We note this is con-

sistent since (10)', like the system schedulability constraints, is also *sustainable* to the periods and deadlines: increasing the periods and deadlines can only make (10)' more satisfiable. In the rest of the section, we slightly abuse the term "schedulability" to also include the utilization bound constraints.

We now detail the procedure in a step-wise manner.

Step 1-Initial Problem. The algorithm first starts with the following optimization problem Π,

$$\begin{aligned} &\min_{\forall \mathcal{R}} d_\Omega \\ s.t.\ &\mathcal{R} \not\preceq \mathcal{U},\ \forall \mathcal{U} \in \mathbb{U} \quad \text{(as formulated in (19))} \\ &L'_p \leq D_p,\ \forall p \\ &(8)' - (9)' \end{aligned} \tag{21}$$

where $\mathbb{U} = \emptyset$ initially (i.e., no MUPDA implied constraints).

Step 2-Solve Problem Π. The second step solves the optimization problem Π. If the Π is infeasible, then the algorithm terminates reporting the infeasibility. This is possible when, for example, the end-to-end deadlines are too tight such that no schedulable solution can satisfy them. Otherwise solving Π returns a solution $\mathcal{R}^*$ that is optimal w.r.t. the current known set of MUPDAs $\mathbb{U}$.

Step 3-$\mathcal{R}^*$ Relaxation. Problem Π only concerns minimizing d_Ω. The values assigned to other virtual deadlines $\langle \tau_i, d_i \rangle^D$ and periods $\langle \tau_i, t_i \rangle^D$ for all $\tau_i \notin \Omega$, though feasible w.r.t. the constraints of Π, may still be arbitrary and unnecessarily small. The consequence is that the resulting solution $\mathcal{R}$ is less likely to be schedulable.

Let the solution obtained from **Step 2** be

$$\mathcal{R}^* = \{\langle \tau_1, d_1^* \rangle^D .. \langle \tau_n, d_n^* \rangle^D, \langle \tau_1, t_1^* \rangle^T .. \langle \tau_n, t_n^* \rangle^T, \langle \Omega, d_\Omega^* \rangle^W\}$$

This step tries to relax $\mathcal{R}^*$ by solving the following problem

$$\begin{aligned} &\max_{\forall \mathcal{R}} \sum_{\forall t_i, d_i \in \mathcal{R}} t_i + d_i \\ s.t.\ &L'_p \leq D_p,\ \forall p \\ &(8)' - (9)' \\ &\mathcal{R} \succeq \mathcal{R}^* \\ &d_\Omega = d_\Omega^* \end{aligned} \tag{22}$$

Intuitively, (22) aims to increase each entry of t_i and d_i in $\mathcal{R}$ while maintaining the objective value d_Ω. Constraint $\mathcal{R} \succeq \mathcal{R}^*$ guarantees that the new solution $\mathcal{R}$ is a relaxation of the original one $\mathcal{R}^*$, hence $\mathcal{R}$ is also an optimal solution to (21). The purpose of the relaxation is to increase the likelihood of termination at Step 2. Generally, the larger t_i and d_i are, the more likely the period and deadline assignment is schedulable.

Step 4. MUPDA Computation. Let $\mathcal{R}$ be the adjusted solution obtained from **Step 3.** If $\mathcal{R}$ is schedulable, then $\mathcal{R}$ is the optimal solution to the full problem (17) and the algorithm terminates. Otherwise, the algorithm computes several MUPDAs and add them to $\mathbb{U}$ (consequently their implied constraints in the form (19) to problem Π). Then it returns to **Step 2.**

In the following, we demonstrate the proposed optimization algorithm using on the example system problem in Table I. The original problem is described in (14). The algorithm first starts with the following initial problem Π

$$\begin{aligned} &\min_{\forall \mathcal{R}} d_\Omega \\ s.t.\ &d_2 + t_2 + d_3 + t_3 \leq 63 \\ &C_i \leq d_i \leq t_i \leq T_i^{\text{ub}},\ \forall \tau_i \end{aligned} \tag{23}$$

which contains only the objective, end-to-end latency constraint, and the bounds on the variables. The $\mathcal{R}$-schedulability constraints are ignored. The algorithm then enters the iteration in **Steps 2–4**.

Iteration 1. Solving the initial problem Π, we get

$$\begin{aligned} \mathcal{R}_1^* = \{ &\langle \tau_1, 10 \rangle^D, \langle \tau_2, 3 \rangle^D, \langle \tau_3, 10 \rangle^D, \langle \tau_4, 100 \rangle^D \\ &\langle \tau_1, 10 \rangle^T, \langle \tau_2, 3 \rangle^T, \langle \tau_3, 10 \rangle^T, \langle \tau_4, 100 \rangle^T, \langle \Omega, 18 \rangle^W \} \end{aligned}$$

The estimated end-to-end latency by the returned solution is

$$d_2 + t_2 + d_3 + t_3 = 26 \tag{24}$$

which is unnecessarily smaller than the required end-to-end latency deadline. Thus the following relaxation is performed on $\mathcal{R}^*$ to increase the period and virtual deadline assignment.

$$\begin{aligned} &\max \sum_{\forall \tau_i} d_i + t_i \\ s.t.\ &d_2 + t_2 + d_3 + t_3 \leq 63 \\ &d_i \leq t_i \leq T_i^{\text{ub}},\ \forall \tau_i \\ &d_2 \geq 3, t_2 \geq 3, d_3 \geq 10, t_3 \geq 10 \\ &d_\Omega = d_\Omega^* \end{aligned} \tag{25}$$

Solving the above problem returns the following adjusted period-deadline assignment.

$$\begin{aligned} \mathcal{R}_1^{'*} = \{ &\langle \tau_1, 10 \rangle^D, \langle \tau_2, 3 \rangle^D, \langle \tau_3, 10 \rangle^D, \langle \tau_4, 100 \rangle^D \\ &\langle \tau_1, 10 \rangle^T, \langle \tau_2, 10 \rangle^T, \langle \tau_3, 40 \rangle^T, \langle \tau_4, 100 \rangle^T, \langle \Omega, 18 \rangle^W \} \end{aligned}$$

While also satisfying the end-to-end deadline constraint, $\mathcal{R}_1^{'*}$ is more relaxed than the original $\mathcal{R}_1^*$ and is more likely to be schedulable. For the rest of the example, we omit the details of relaxation and only show the adjusted solution after relaxation.

$\mathcal{R}_1^{'*}$ is not schedulable. The following two MUPDAs are computed using Algorithm 1.

$$\begin{aligned} \mathcal{U}_1 = \{ &\langle \tau_1, 10 \rangle^D, \langle \tau_2, 20 \rangle^D, \langle \tau_3, 40 \rangle^D, \langle \tau_4, 100 \rangle^D \\ &\langle \tau_1, 10 \rangle^T, \langle \tau_2, 20 \rangle^T, \langle \tau_3, 40 \rangle^T, \langle \tau_4, 100 \rangle^T, \langle \Omega, 34 \rangle^W \} \\ \mathcal{U}_2 = \{ &\langle \tau_1, 10 \rangle^D, \langle \tau_2, 20 \rangle^D, \langle \tau_3, 19 \rangle^D, \langle \tau_4, 100 \rangle^D \\ &\langle \tau_1, 10 \rangle^T, \langle \tau_2, 20 \rangle^T, \langle \tau_3, 40 \rangle^T, \langle \tau_4, 100 \rangle^T, \langle \Omega, 43 \rangle^W \} \end{aligned}$$

$\mathcal{U}_1$ implies the following constraint in the form of (19)

$$\begin{aligned} &(d_1 > 10) \vee (d_2 > 20) \vee (d_3 > 40) \vee (d_4 > 100) \vee \\ &(t_1 > 10) \vee (t_2 > 20) \vee (t_3 > 40) \vee (t_4 > 100) \vee (d_\Omega > 34) \end{aligned}$$

where $\vee$ is another way to represent logical OR operations.

Taking into consideration the bounds and integrality of variables, The above constraint can be simplified as

$$d_\Omega \geq 35 \tag{26}$$

Similarly, $\mathcal{U}_2$ implies the following constraint

$$(d_3 \geq 20) \vee (d_\Omega \geq 44) \tag{27}$$

The above two constraints are added to problem Π.

Iteration 2. The updated problem Π has the following solution.

$$\begin{aligned}\mathcal{R}_2^* =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,3\rangle^D,\langle\tau_3,20\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,3\rangle^T,\langle\tau_3,37\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,170\rangle^W\}\end{aligned}$$

$\mathcal{R}_2^*$ is not schedulable. The following two MUPDAs are computed.

$$\begin{aligned}\mathcal{U}_3 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,20\rangle^D,\langle\tau_3,40\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,19\rangle^T,\langle\tau_3,40\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,39\rangle^W\}\\ \mathcal{U}_4 =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,4\rangle^D,\langle\tau_3,40\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,40\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,35\rangle^W\}\end{aligned}$$

The following MUPDA implied constraints are added to Π.

$$((t_2 \geq 20) \vee (d_\Omega \geq 40)) \wedge ((d_2 \geq 5) \vee (d_\Omega \geq 36))$$

where $\wedge$ represents the logical AND operation.

Iteration 3. The updated problem Π allows the solution below.

$$\begin{aligned}\mathcal{R}_3^* =&\{\langle\tau_1,10\rangle^D,\langle\tau_2,3\rangle^D,\langle\tau_3,20\rangle^D,\langle\tau_4,100\rangle^D\\ &\langle\tau_1,10\rangle^T,\langle\tau_2,20\rangle^T,\langle\tau_3,20\rangle^T,\langle\tau_4,100\rangle^T,\langle\Omega,36\rangle^W\}\end{aligned}$$

$\mathcal{R}_3^*$ is now schedulable, which gives the optimal period assignment. The corresponding schedulable priority assignment, which is returned from Algorithm 3 with $\mathcal{R}_3^*$ as the input, is $\tau_2 \succ \tau_1 \succ \tau_4 \succ \tau_3$. The optimal objective is $d_\Omega = 36$.

Remark 2. Each MUPDA implied constraint introduces additional binary variables for modeling disjunction. For example, consider $(d_3 \geq 20) \vee (d_\Omega \geq 44)$. It can be formulated in ILP as

$$\begin{aligned}&(d_3 \geq 20b_1) \wedge (d_\Omega \geq 44b_2)\\ &b_1 + b_2 \geq 1\end{aligned} \tag{28}$$

where b_1 and b_2 are binary variables. A MUPDA $\mathcal{U}$ may introduce $|\mathcal{U}|$ number of variables in the worst case. Thus, given a set of MUPDAs $\mathbb{U}$, the number of additional binary variables for the ILP problem is $O(\sum_{\forall\mathcal{U}\in\mathbb{U}}|\mathcal{U}|)$.

As the total number of MUPDAs for a system is bounded, the proposed procedure in Figure 1 is guaranteed to terminate. Upon termination, the procedure either returns an optimal solution if the problem is feasible, otherwise it reports infeasibility. This is because in each iteration, only a subset of the MUPDA implied constraints is added into the problem Π, hence Π maintains to be a relaxation of the original problem. An optimal and schedulable solution to Π must also be an optimal solution of the original problem.

VI. Experimental Results

In this section, we present the experimental results on two industrial case studies. We compare our approach with the state-of-the-art method for period optimization that is based on a MIGP formulation or a GP-based iterative procedure [7]. All runtimes are the wall-clock time on a dedicated machine with a 2.5GHz eight-core processor and 8GB memory. Since the previous approaches only handle the period optimization problem where the priority assignment is given, we consider this version for a direct comparison. Then we us problem where priority assignment is also part of the decision variables to show the benefit of our unified framework.

A. Vehicle with Active Safety Features

Our first case study consists of an industrial experimental vehicle system with active safety features [7]. The system contains 29 Electronic Control Units (ECUs) connected through 4 CAN buses. A total of 92 tasks are deployed on the ECUs and 192 CAN messages are exchanged on the CAN buses. Tasks are scheduled preemptively and messages are non-preemptive. End-to-end deadlines are imposed on 12 pairs of source-sink tasks, between which a total of 222 unique end-to-end paths exist. 9 pairs of communicating tasks on the same ECU are imposed with period harmonicity requirements. The total utilization of each execution platform must not exceed 70% for future extensibility.

The allocation of tasks to ECUs and messages to CAN buses is given. Worst case execution time of each object is also measured. An initial assignment of periods and priorities is given by the designer, which fails to satisfy any of the end-to-end deadline constraints. The problem is then to find a new feasible assignment that meets schedulability and end-to-end deadline constraints. Different from [7], the initial period assignments are used as the upper bound T_i^{ub} of the period variable T_i.

Optimization of Period Assignment. To give a direct comparison with the approaches in [7], we first consider the optimization of period assignment with given priorities, and the objective is to optimize the WCRT summation over all objects. Since [7] can only handle the response time analyses in (2) and (7), we assume tasks/messages have implicit deadline and adopt the same analyses. We first fix the harmonicity factors to the value initially given by the designer.

We try to solve the MIGP formulation proposed in [7] by the BnB solver in YALMIP [17] which leverages gpposy [20] to solve geometric programming problems. However, YALMIP always reports "out of memory" before finding any feasible solution.

We now compare our approach (called `MUPDA-guided` later) with the iterative geometric programming based algorithm (`IterGP`) proposed in [7], as well as genetic algorithm (GA). The main idea of `IterGP` is to approximate the response time analysis, i.e., (2) as that of (13), which allows to formulate the problem as a geometric program. When the approximated response time R_i' is different from the actual one R_i, the parameter $\alpha_{i,j}$ is updated to try to reduce the approximation error, after which the optimization is performed again. The algorithm terminates when the maximum approximation error is within an acceptable bound or the iteration limit is reached. As in [7], we set the maximum iteration limit to be 15 (which is sufficient as `IterGP` usually gets stuck at local minimum before that). The algorithm is implemented in YALMIP framework [17] using the gpposy geometric programming solver [20].

TABLE II: Optimization results for the experimental vehicle with given priority assignment

Method	Objective	Time	Status
MUPDA-guided	541708	24.35s	Terminate
IterGP	541767	185.68s	Iteration Limit Reached
GA-10^5	N/A	6.3h	Abort
GA-10^6	N/A	$\geq 48h$	Timeout

TABLE III: Optimization results for the experimental vehicle with given priority assignment, relaxed harmonicity factor

Method	Objective	Time	Status
MUPDA-guided	532938	6.43s	Terminate
IterGP	533770	999.54s	Iteration Limit Reached

We implement the GA-based approach leveraging the MATLAB optimization toolbox. The objective is set as the fitness function, and system schedulability and end-to-end deadline requirement are provided as nonlinear constraints. We use two initial population sizes 10^5 and 10^6, denoted as GA-10^5 and GA-10^6 respectively. All other parameter settings are the default values in MATLAB.

The results are summarized in Table II. MUPDA-guided terminates with the optimal solution after around 24 seconds. IterGP terminates with a slightly sub-optimal solution after exceeding the limit of 15 iterations. GA-10^5 aborts after failing to find any feasible solution for three generations. GA-10^6 on the other hand, is unable to complete the first generation within 48 hours. Figure 2 plots the objective value during the optimization process of IterGP. Each data point in the plot corresponds to the objective value of a particular iteration. The curve "Approximated GP Objective" corresponds to the objective value of the geometric optimization, which is given by the approximated response time (13). The "Actual Objective" corresponds to the objective value by the actual response time (2). If the period assignment is actually infeasible (w.r.t. schedulability and end-to-end deadline constraints), the corresponding data point is omitted in the figure, which is why for some data points on the "Approximated GP Objective" curve, there is no corresponding one on the "Actual Objective" curve. As shown in the figure, the solution summarized in Table II is found by IterGP after the third iteration at time 39s. However, after that IterGP oscillates between a feasible and an infeasible period assignments, and cannot find any better solution. This highlights a major drawback of IterGP: unlike MUPDA-guided, IterGP cannot guarantee convergency.

Next, we relax the harmonicity factor $h_{i,j}$ from a fixed constant to an integer decision variable. We omit GA for this setting since the *ga* function provided in MATLAB is not capable of handling equality constraints with integer variables. The corresponding results are summarized in Table III. MUPDA-guided finds optimal solution after only around 6 seconds. IterGP again starts oscillating between a feasible and infeasible solution after a few iterations. The best solution it finds is at the first iteration after 52 seconds.

Optimization of Both Period and Priority. One advantage of MUPDA-guided is its ability to optimize both periods and priority assignments. In this experiment, we ignore the given priority assignment and consider it to be also part of the decision variables. All other experimental settings remains the same as the previous one. Since the approaches in [7] (IterGP and MIGP) are no longer applicable, we evaluate only the proposed technique MUPDA-guided.

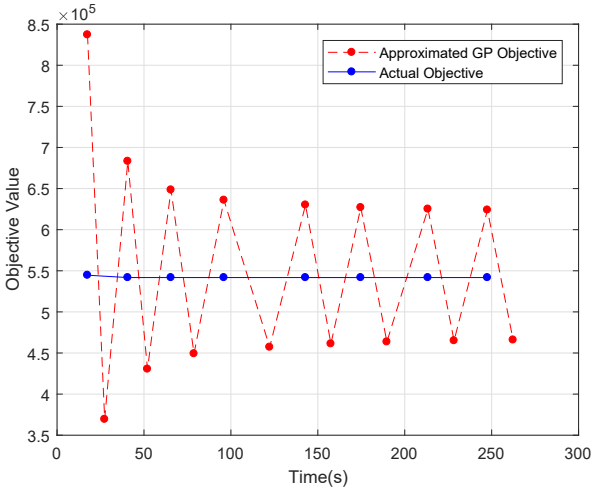


Fig. 2: Optimized objective of IterGP.

TABLE IV: Optimization results for the experimental vehicle without given priority assignment

Harmonicity Factor	Objective	Time	Status
Fixed	300156	14.09s	Terminate
Relaxed	298492	9.80s	Terminate

Table IV summarizes the optimization results with fixed and relaxed harmonicity factor settings. Comparing with the results in Table II and Table III, the inclusion of priority assignment into the decision space significantly improves the optimization results: for fixed harmonicity factor the improvement is about 44.6%, and for relaxed harmonicity factor the objective is about 44.0% smaller. This is expected as the design becomes much more flexible.

In summary, the results in Tables II–IV demonstrate the three advantages of MUPDA-guided compared to IterGP: (i) it guarantees convergency; (ii) it may run magnitudes faster (e.g., 9.80s vs. 999.54s for the setting of relaxed harmonicity factor); (iii) it can handle the co-optimization of both periods and priority assignments, hence has the potential to provide substantially better solutions.

Optimization with Arbitrary Deadline Setting. Another advantage of the proposed algorithm is its capability of accommodating schedulability analysis that may be difficult or even impossible to use in standard mathematical programming framework, such as those in (3) and (6) for arbitrary deadline setting. In this experiment, we seek to evaluate the benefit brought by this feature. Intuitively, larger deadline allows more flexibility in system design in the sense that more priority assignments would be considered schedulable compared to

TABLE V: Optimization results for relaxed deadline settings

Implicit Deadline			
Objective		Time	Status
134502		6.12s	Terminate
Relaxed Deadline			
ρ	Objective	Time	Status
1	134502	6.06s	Terminate
2	128972	9.49s	Terminate
3	120832	7.36s	Terminate

constrained or implicit deadline settings. This would be beneficial, for example, when the design objective only involves a subset of objects and other objects may be scheduled at lower priority levels when given longer deadlines. In the following, we optimize a variant of the original problem as follows.

• Instead of the sum of WCRTs over all objects, we now optimize the sum over only those objects in the end-to-end paths (i.e., Ω is the set of objects in the end-to-end paths).

• The set of objects in Ω still uses implicit deadline setting. Other objects use the more relaxed arbitrary deadline setting with response time analysis in (3) and (6). The deadline of an object τ_i not in Ω is set to ρT_i^{init}, where T_i^{init} is the initial period assignment for τ_i from the designer, and ρ is a scaling factor. The higher the ρ, the more relaxed the deadline constraint is.

Other settings remain the same with the previous experiment on optimizing both period and priority. For simplicity, only fixed harmonicity factor setting is considered.

Table V summarizes the results of `MUPDA-guided` for the baseline case that all tasks/messages have implicit deadline, as well as the setting with relaxed deadlines for tasks/messages not in the end-to-end paths. Intuitively, the modified problem places more emphasis on the set of objects in Ω: the higher the priority that can be assigned to objects in Ω, the smaller the objective value.

As shown in the table, the use of relaxed deadline setting does bring improvement of optimization results. The larger the ρ value, the smaller the objective value. Intuitively, when objects not in Ω are given relaxed deadlines, they are able to tolerate lower priorities, which makes it possible to schedule objects in Ω at higher priorities. This reduces the sum of WCRTs over Ω. However, this requires to use sophisticated analyses in (3) and (6) that is very difficult in standard mathematical programming framework.

B. Distributed System with Redundancy based Fault-Tolerance

Our second case study is an example system used in [24]. The system is designed in a fault-tolerant manner by replicating tasks in the original design onto different ECUs, which results in a total of 43 tasks and 36 messages deployed onto an architecture with 8 ECUs. However, for merely the purpose of period optimization, we do not distinguish between original and replicated tasks and simply treat all of them as different periodic tasks in a normal periodic task system. Initial allocation, period assignments for tasks are given, and the initial period of each message is assumed to be the same as its source task. Tasks are preemptive and messages are non-preemptive, with an initial priority assignment that follows the rate-monotonic policy. End-to-end deadlines are imposed on 6 paths, which are assumed to be the initial end-to-end latency on the path. There is no harmonicity constraint. The utilization bound of each ECU and bus is set to 70%.

TABLE VI: Optimization results for the fault-tolerant system with given priority assignment

Method	Objective	Time	Status
`MUPDA-guided`	50656	0.29s	Terminate
`IterGP`	51600	9.3s	Iteration Limit Reached
`MIGP`	50656	13.27s	Terminate

This case study is noticeably smaller than the previous one on the experimental vehicle, which allows the MIGP formulation for optimizing period to be solved by YALMIP. Table VI summarizes the results on the three methods (`MUPDA-guided`, `IterGP`, and `MIGP`) for optimizing the period under the given rate-monotonic priority assignment. Again, `IterGP` oscillates and has to settle for a suboptimal solution after 15 iterations. `MUPDA-guided` and `MIGP` both find the optimal solution but `MUPDA-guided` is evidently faster. If we also include the priority assignment as the decision variable, `MUPDA-guided` is the only one capable of solving this problem. It finds a much better solution with an objective of 42740 in 0.51 second, due to the additional design space of priority assignment.

VII. Conclusion

This paper considers the problem of automating the period and priority assignment stage in the design of distributed hard real-time systems. Such problems are common in a wide variety of embedded systems application domains such as automotive and avionics. Our approach is to develop a customized optimization procedure, that leverages the strength of ILP solver and problem-specific algorithms. Compared to the state-of-the-art work for solving a subproblem of optimizing period assignment only, the proposed algorithm runs much faster while providing substantially better solutions.

Acknowledgments

This work is in part supported by National Science Foundation of USA (Grant No. 1812963).

References

[1] N. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39 – 44, 2001.

[2] S. Baruah and A. Burns. Sustainable scheduling analysis. In *IEEE Real-Time Systems Symposium*, 2006.

[3] I. Bate and P. Emberson. Incorporating scenarios and heuristics to improve flexibility in real-time embedded systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2006.

[4] E. Bini and A. Cervin. Delay-aware period assignment in control systems. In *IEEE Real-Time Systems Symposium*, 2008.

[5] E. Bini and M. Di Natale. Optimal task rate selection in fixed priority systems. In *IEEE Real-Time Systems Symposium*, 2005.

[6] Y. Chu and A. Burns. Flexible hard real-time scheduling for deliberative ai systems. *Real-Time Systems*, 40(3):241–263, 2008.

[7] A. Davare, Q. Zhu, M. Di Natale, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli. Period optimization for hard real-time distributed automotive systems. In *Design Automation Conference*, 2007.

[8] R. Davis and A. Burns. Robust priority assignment for fixed priority real-time systems. In *IEEE Real-Time Systems Symposium*, 2007.

[9] R. Davis and A. Burns. Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In *IEEE Real-Time Systems Symposium*, 2009.

[10] R. Davis, A. Burns, R. Bril, and J. Lukkien. Controller area network schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[11] R. Davis, L. Cucu-Grosjean, M. Bertogna, and A. Burns. A review of priority assignment in real-time systems. *J. Syst. Archit.*, 65(C):64–82, Apr. 2016.

[12] P. Deng, Q. Zhu, A. Davare, A. Mourikis, X. Liu, and M. Di Natale. An efficient control-driven period optimization algorithm for distributed real-time systems. *IEEE Transactions on Computers*, 65(12):3552–3566, 2016.

[13] M. Di Natale, P. Giusto, S. Kanajan, C. Pinello, and P. Popp. Optimizing end-to-end latencies by adaptation of the activation events in distributed automotive systems. In *Society of Automotive Engineers World Congress*, 2007.

[14] M. Di Natale, L. Guo, H. Zeng, and A. Sangiovanni-Vincentelli. Synthesis of multi-task implementations of simulink models with minimum delays. *IEEE Trans. Industrial Informatics*, 6(4):637–651, 2010.

[15] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson. A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics. In *IEEE Real-Time Systems Symposium*, 2009.

[16] A. Hamann, M. Jersak, K. Richter, and R. Ernst. Design space exploration and system optimization with symta/s - symbolic timing analysis for systems. In *IEEE Real-Time Systems Symposium*, 2004.

[17] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *IEEE Symposium on Computer Aided Control Systems Design*, 2004.

[18] G. Mancuso, E. Bini, and G. Pannocchia. Optimal priority assignment to control tasks. *ACM Trans. Embedded Computing Systems*, 13(5s):161, 2014.

[19] MathWorks. *The MathWorks Simulink and StateFlow User's Manuals*. [Online] http://www.mathworks.com.

[20] A. Mutapcic, K. Koh, S. Kim, and S. Boyd. Ggplab version 1.00: a matlab toolbox for geometric programming, January 2006.

[21] M. Oral and O. Kettani. A linearization procedure for quadratic and cubic mixed-integer problems. *Operations Research*, 40(1-supplement-1):S109–S116, 1992.

[22] M. Saksena and Y. Wang. Scalable real-time system design using preemption thresholds. In *IEEE Real-Time Systems Symposium*, 2000.

[23] D. Seto, J. P. Lehoczky, and L. Sha. Task period selection and schedulability in real-time systems. In *IEEE Real-Time Systems Symposium*, 1998.

[24] K. Tindell, A. Burns, and A. Wellings. Allocating hard real-time tasks: An np-hard problem made easy. *Real-Time Syst.*, 4(2):145–165, 1992.

[25] C. Wang, Z. Gu, and H. Zeng. Global fixed priority scheduling with preemption threshold: Schedulability analysis and stack size minimization. *IEEE Trans. Parallel and Distributed Systems*, 27(11):3242–3255, Nov. 2016.

[26] Y. Wang and M. Saksena. Scheduling fixed-priority tasks with preemption threshold. In *IEEE Conf. Real-Time Computing Systems and Applications*, 1999.

[27] E. Wozniak, M. Di Natale, H. Zeng, C. Mraidha, S. Tucci-Piergiovanni, and S. Gerard. Assigning time budgets to component functions in the design of time-critical automotive systems. In *ACM/IEEE International Conference on Automated Software Engineering*, 2014.

[28] H. Zeng and M. Di Natale. An efficient formulation of the real-time feasibility region for design optimization. *IEEE Trans. Computers*, 62(4):644–661, Apr. 2013.

[29] H. Zeng, M. Di Natale, and Q. Zhu. Minimizing stack and communication memory usage in real-time embedded applications. *ACM Trans. Embed. Comput. Syst.*, 13(5s):1–25, July 2014.

[30] H. Zeng, A. Ghosal, and M. D. Natale. Timing analysis and optimization of flexray dynamic segment. In *10th IEEE International Conference on Computer and Information Technology*, pages 1932–1939, June 2010.

[31] H. Zeng and M. D. Natale. Efficient implementation of autosar components with minimal memory usage. In *IEEE International Symposium on Industrial Embedded Systems*, pages 130–137, June 2012.

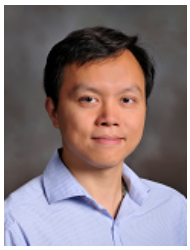
[32] Y. Zhao and H. Zeng. The concept of unschedulability core for optimizing priority assignment in real-time systems. In *Conference on Design, Automation and Test in Europe*, 2017.

[33] Y. Zhao and H. Zeng. The virtual deadline based optimization algorithm for priority assignment in fixed-priority scheduling. In *IEEE Real-Time Systems Symposium*, 2017.

[34] Y. Zhao and H. Zeng. The concept of response time estimation range for optimizing systems scheduled with fixed priority. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2018.

Yecheng Zhao Yecheng Zhao is currently pursuing the PhD degree in Computer Engineering at Virginia Tech. He received his B.E. in Electrical Engineering from Harbin Institute of Technology, Harbin, China. His main research interest is design optimization for real-time embedded systems.

Vinit Gala Vinit Gala is currently working as a Technical Solutions Engineer at Google Inc. He received his M.Eng. in Computer Engineering from Virginia Tech and B.E. in Electronics and Telecommunication from Mumbai University, India. His research interest is System Software with emphasis on real-time systems.



Haibo Zeng Haibo Zeng is currently a faculty member at Virginia Tech. He received his Ph.D. in Electrical Engineering and Computer Sciences from University of California at Berkeley, and B.E. and M.E. in Electrical Engineering from Tsinghua University, Beijing, China. He was a senior researcher at General Motors R&D until October 2011, and an assistant professor at McGill University, Canada until August 2014. His research interests are embedded systems, cyber-physical systems, and real-time systems, with four best paper/best student paper awards in the above fields.