Optimal harmonic period assignment

complexity results and approximation algorithms

Mohaqeqi, Morteza; Nasri, Mitra; Xu, Yang; Cervin, Anton; Årzén, Karl Erik

Published in:
Real-Time Systems

DOI:
10.1007/s11241-018-9304-0

2018

Document Version:
Peer reviewed version (aka post-print)

Real-Time Syst manuscript No.
(will be inserted by the editor)

Optimal Harmonic Period Assignment: Complexity Results and Approximation Algorithms

Morteza Mohaqeqi · Mitra Nasri · Yang Xu · Anton Cervin · Karl-Erik Årzén

Received: date / Accepted: date

Abstract Harmonic periods have wide applicability in industrial real-time systems. Rate monotonic (RM) is able to schedule task sets with harmonic periods up to 100% utilization. Also, if there is no release jitter and execution time variation, RM and EDF generate the same schedule for each instance of a task. As a result, all instances of a task are interfered by the same amount of workload. This property decreases the jitters that happen during sampling and actuation of the tasks, and hence, it increases the quality of service in control systems. In this paper, we consider the problem of optimal period assignment where the periods are constrained to be harmonic and the task set is required to be feasible. We study two variants of this problem. In the first one, the objective is to maximize the system utilization, while in the second one, the goal is to minimize the total weighted sum of the periods. First, we assume that an interval is determined *a priori* for each task from which its period can be selected. We show that both variants of the problem

This work has been supported by the Alexander von Humboldt Foundation and by the ELLIIT Excellence Center at Linköping–Lund.

M. Mohaqeqi
Department of Information Technology, Uppsala University, Sweden
E-mail: morteza.mohaqeqi@it.uu.se

M. Nasri
Max Planck Institute for Software Systems (MPI-SWS), Germany
E-mail: mitra@mpi-sws.org

Y. Xu
Department of Automatic Control, Lund University, Sweden
E-mail: yang@control.lth.se

A. Cervin
Department of Automatic Control, Lund University, Sweden
E-mail: anton@control.lth.se

K.-E. Årzén
Department of Automatic Control, Lund University, Sweden
E-mail: karlerik@control.lth.se

are (at least) weakly NP-hard. This is shown by reducing the NP-complete number partitioning problem to the mentioned harmonic period assignment problems. Afterwards, we consider a variant of the second problem in which the periods are not restricted to a special interval. We present two approximation algorithms with polynomial-time complexity for this problem and show that the maximum relative error of these algorithms is bounded by a factor of 1.125. Our evaluations show that, on the average, results of the approximation algorithms are very close to an optimal solution.

Keywords Harmonic Tasks · Period Assignment · Real-Time Schedulability · Hard Real-Time

1 Introduction

Selecting appropriate timing parameters for the tasks such that performance objectives and design constraints are met is an important step in the design of real-time systems. Specifically, in the real-time systems with *periodic* tasks, selecting suitable periods influences a number of prominent properties including the system schedulability [1, 2], jobs' response times [3], and the related jitters [4]. Harmonic periods, namely the periods that pairwise divide each other [5], exhibit specific characteristics which help the designers to achieve better solutions with respect to the mentioned properties. For instance, schedulability analysis of the task sets with constrained-deadlines can be done efficiently (i.e., in polynomial time) when the periods are harmonic [6]. This is particularly important because in the general case the problem is hard for both fixed-priority [7] and dynamic-priority [8] scheduling policies. Furthermore, while the uniprocessor schedulable utilization of the fixed-priority scheduling policy can be lower than 70% [9] for some periodic task sets, the respective value for the harmonic task sets is 100% [5].

Harmonic periods have been widely used in industrial applications ranging from radar dwell tasks [14] and robotics [15, 17–19] to control systems with nested feedback loops [20]. If periods are harmonic, it will be possible to apply optimal fault tolerant mechanisms as mentioned in [21]. In the integrated modular avionics (IMA), harmonic periods facilitate the problem of assigning tasks to the partition servers [31]. Consequently, they reduce the size of the hyperperiod which facilitates the construction and storage of the offline schedule table used in IMA systems because then a smaller portion of RAM (or flash memory) of the system is consumed by the offline table. It is an important requirement for the systems with a limited memory such as Atmel UC3A0512 microcontroller that is used for mission critical space applications [16].

Because of the mentioned advantages, finding harmonic periods for a given task set which satisfy performance requirements in the system is an important problem for many real-time systems. In [5] two algorithms called *Sr* and *DCT* have been introduced to find the largest harmonic periods which are pairwise smaller than a given set of periods. These algorithms were later used in period assignment for radar tasks [14]. However, they are not designed to tackle with

period ranges or to find the smallest harmonic periods with a feasible utilization. In [10] and [11] two pseudo-polynomial time algorithms are proposed to verify the existence of a harmonic period assignment for a given set of period ranges. However, the essential complexity of the problem of finding the harmonic periods such that a design objective, such as the total utilization, is optimized is not explored. In a recent work, Xu et al. [29] has proposed an optimal solution to find a set of harmonic periods that have the minimum total distance from a given set of periods. The proposed solution, however, has exponential computational complexity.

In the domain of control systems, Eker et al., [25] have proposed a method to assign the period of each task such that the task set utilization remains under a certain value and a cost function that reflects the effect of task's period on the control cost is minimized. The proposed solution, however, is limited to the cost functions that are convex. The problem of minimizing control cost using an online or offline period assignment method has been considered by Henriksson et al. [26] and Bini et al. [22], respectively. However, these approaches neither try nor guarantee that the resulting periods are harmonic.

In this paper, we discuss the computational complexity of finding feasible harmonic periods for a given set of real-time tasks where the goal is to maximize the total utilization while the periods must be selected from a given set of intervals. We show that the problem is NP-hard by transforming the well-known number partitioning problem to the harmonic period assignment problem. We establish the same complexity result for a variant of the problem in which the goal is to minimize the weighted sum of the periods. Additionally, we present two polynomial-time approximation algorithms for a variant of the second problem in which the periods are not bounded to any interval. We derive a tight bound of 1.125 for the maximum error of our algorithms with respect to an optimal algorithm. Simulation results show that, on average, this error is smaller than 1.125, which means that the approximation algorithm behaves very close to the optimal one. It is worth noting that the complexity of this variant of the problem, i.e., the one with unrestricted periods, is still unknown.

Our results can be used in the design of control systems in which the control performance is expressed (or can be approximated) by a linear function of the periods. In these systems, jitters in the sampling and actuation can adversely affect the quality of control (QoC) [22]. These jitters can be efficiently reduced if the control tasks use harmonic periods because then each time the task is released, the same set of high priority tasks are released in the system. In this paper, we evaluate our period assignment solution for control systems and compare the results with an optimal non-harmonic period assignment.

This paper is an extended version of "On the Problem of Finding Optimal Harmonic Periods" by Mohaqeqi et al., [30] that was published in RTNS 2016. The main additional contributions are as follows. In this paper, we establish the computational complexity of the problem of finding feasible harmonic periods from a set of given period ranges, where the goal is to minimize the weighted sum of the periods. In the RTNS paper, the hardness of finding fea-

sible harmonic periods was studied only for the case of maximizing the total system utilization. Moreover, in the previous paper, the approximation error bound that we had for our approximation solutions was 2 while in this paper we have derived the tight error bound of our algorithms and show that this bound is $\frac{9}{8} = 1.125$. Finally, we have added more experiments and considered the effect of more parameters on the relative error of our algorithms, e.g., the effect of weight of the periods in the goal function of the optimization. Furthermore, in order to assess the complexity and scalability of the proposed methods in practice, we have reported the average number of operations required by each algorithm to find harmonic periods. We have also expanded control evaluations, considering more realistic plants and real-time tasks with variable execution times.

The reminder of the paper is organized as follows; Sect. 2 introduces the notations and formally describes the considered harmonic period assignment problems. The complexity proofs are presented in Sect. 3. Next, the approximation algorithms for finding harmonic periods are presented in Sect. 4 and are evaluated in Sect. 5. A summary of the paper and a discussion on the related open problems are given in Sect. 6. Finally, the conclusion and future work are presented in Sect. 7.

2 Notations and Definitions

We consider a set of n real-time tasks $\tau_1, \ldots, \tau_n$ with the set of worst-case execution times (WCETs) $\mathbf{c} = \{C_1, C_2, \ldots, C_n\}$. Further, for each task τ_i, an interval $I_i = [I_i^s, I_i^e]$ is given as the period range. The period of each task τ_i, which is denoted by T_i, must be selected from I_i. The utilization of τ_i is defined as $U_i = C_i/T_i$. In addition, the total utilization of the system is defined as the sum of all utilizations: $U = \sum_{i=1}^{n} U_i$. We use $\mathbb{N}$ to denote the set of positive integers.

A set of periods is said to be harmonic if the pairwise periods divide each other. More specifically, the set of periods $\{T_1, T_2, \ldots, T_n\}$ is harmonic if for every T_i and T_j, either $T_i/T_j \in \mathbb{N}$, or $T_j/T_i \in \mathbb{N}$. A set of harmonic tasks is feasible if and only if $U \leq 1$. While the period ratios of harmonic tasks are required to be integer, it is assumed that the values of individual periods are not restricted to be integer.

The goal is to assign a set of harmonic periods to the task set such that some design objective is optimized. For example, for control applications, the objective could be to assign the smallest feasible periods [22]. In this paper, we consider two harmonic period assignment problems as specified below.

Problem 1 Take a set of n real-time tasks with the given WCETs and period ranges. The goal is to assign a period to each task such that the total utilization is maximized, while the periods are harmonic and the task set is feasible.

Formally, the problem is specified as

$$\text{maximize } U = \sum_{1 \le i \le n} U_i \tag{1a}$$

subject to :

$$T_i \in I_i, \qquad \text{for } 1 \le i \le n \tag{1b}$$

$$\frac{T_i}{T_j} \in \mathbb{N} \text{ or } \frac{T_j}{T_i} \in \mathbb{N}, \qquad \text{for } 1 \le i, j \le n \tag{1c}$$

$$U \le 1. \tag{1d}$$

In the subsequent sections, we refer to this problem as the utilization-maximizing harmonic period assignment (UHPA) problem. This problem targets finding a feasible (and schedulable) set of harmonic periods which allows highly utilizing the resource. Regarding existing solutions to this problem, two pseudo-polynomial algorithms to verify the existence of a harmonic assignment are already introduced in [10] and [11] though none of them guarantee that the resulting assignment has $U \le 1$. In the current work, we try to shed some light on the inherent complexity of the problem.

In the second problem, the goal is to minimize the weighted sum of the selected periods, as defined below.

Problem 2 Consider a set of n tasks with given WCETs and period intervals. Then, the problem is specified as follows.

$$\text{minimize } \sum_{1 \le i \le n} w_i T_i \tag{2a}$$

subject to :

$$T_i \in I_i, \qquad \text{for } 1 \le i \le n \tag{2b}$$

$$\frac{T_i}{T_j} \in \mathbb{N} \text{ or } \frac{T_j}{T_i} \in \mathbb{N}, \qquad \text{for } 1 \le i, j \le n \tag{2c}$$

$$U \le 1, \tag{2d}$$

where w_i determines how much T_i contributes to the total sum. It is supposed that w_i is given for each task τ_i.

In the following, this problem is referred to as the cost-minimizing harmonic period assignment (CHPA) problem. CHPA can be used in the design of control systems in which the control performance is expressed as a linear function of the periods. While more complex metrics could lead to better results, a weighted sum of periods is a simple approximation, which is also used in the literature. For instance, in [23], it is argued that linear cost functions are reasonable approximations for plants that are sampled reasonably fast.

Moreover, if periods are not harmonic (even if there are harmonic groups), the jobs' response time may vary, and then, (i) calculating the response-time jitter becomes an issue (see [7]), and, (ii) the cost function now also depends

on the jitter, making the overall design problem harder. In contrast, for those applications in which execution times are constant, in the described problems, jitters are forced to be 0 by forcing the periods to be harmonic. Thus, it both allows more accurate description of the controller cost function and less jitter.

In this paper, we provide a lower bound on the computation complexity of CHPA. Also, we propose two approximation algorithms for this problem, when Constraint (2b) is relaxed, that is, when $I_i = [0, \infty]$, for $1 \leq i \leq n$. We use CHPA$^\infty$ to denote this relaxed version of the problem.

3 Complexity Results

In this section, we study the computational complexity of UHPA and CHPA problems. In order to show the hardness of these problems, we present a polynomial time algorithm for reducing any given instance of the *number partitioning* (PART) problem to an instance of these problems. We first review the PART problem. Then, the transformation method for each problem is described.

3.1 Number Partitioning Problem

This section reviews the number partitioning (PART) problem which is known as a (weakly) NP-complete problem[1].

Definition 1 (Number Partitioning (PART)) Let $A = \{a_1, \ldots, a_n\}$ be a set of n items with an associated *size* function $s : A \mapsto \mathbb{N}$ which assigns a positive integer to each item. The problem is to determine whether A can be partitioned into two sets A_1 and A_2 such that the total size of items in A_1 equals that of A_2. More formally, let S, S_1, and S_2 denote the sum of items size for A, A_1, and A_2, respectively. That is,

$$S = \sum_{a_i \in A} s(a_i) \tag{3}$$

$$S_1 = \sum_{a_i \in A_1} s(a_i) \tag{4}$$

$$S_2 = \sum_{a_i \in A_2} s(a_i) \tag{5}$$

Then, the problem is to decide whether A can be partitioned into A_1 and A_2 (i.e., $A_1 \cup A_2 = A$ and $A_1 \cap A_2 = \emptyset$) such that $S_1 = S_2$. An instance of this problem is said to be a *positive* one if such a partitioning exists.

Theorem 1 (Hardness of the PART Problem [13]) *The PART problem is NP-complete. However, it can be solved in pseudo-polynomial time.*

[1] A problem is weakly NP-complete if it is NP-complete and it has a pseudo-polynomial time solution.

3.2 Complexity of UHPA

For the complexity analysis, we present a polynomial time method for transforming any given instance of the PART problem to an instance of UHPA. We show that an instance of PART is a positive one (i.e., the set A can be partitioned to the desired sets A_1 and A_2) if and only if the corresponding UHPA problem has a solution with the total utilization of one. The method is specified in the following.

Consider an instance of the PART problem as specified in Definition 1. The corresponding UHPA problem is specified by a set of $n + 2$ real-time tasks. The WCET of τ_i, for $1 \leq i \leq n$, is determined as

$$C_i = \frac{4s(a_i)}{3S + 3}. \tag{6}$$

In addition, we set $I_i = [1, 2]$ as the interval from which the task period can be selected. Also, for τ_{n+1} and τ_{n+2} we choose

$$C_{n+1} = C_{n+2} = \frac{2}{3S + 3}, \tag{7}$$

$I_{n+1} = [1, 1]$ and $I_{n+2} = [2, 2]$.

Lemma 1 *A given instance of the PART problem is positive (i.e., the given set can be partitioned) if and only if the UHPA problem instance obtained from the above-mentioned transformation method has a solution with $U = 1$.*

Proof Let T_i denote the period of task τ_i assigned by a solution to the UHPA problem. According to the specified UHPA problem, the period of τ_{n+1} and τ_{n+2}, namely T_{n+1} and T_{n+2}, are forced to be 1 and 2, respectively. Further, for the period of tasks τ_i, for $1 \leq i \leq n$, there are only two options, i.e. 1 and 2 (otherwise, $T_i/T_{n+1} \notin \mathbb{N}$ and $T_{n+1}/T_i \notin \mathbb{N}$, which violates constraint (1c)). A schematic view of a sample period assignment is shown in Fig. 1. We define J_1 and J_2 as

$$J_1 = \{i \mid T_i = 1, 1 \leq i \leq n\}$$

$$J_2 = \{i \mid T_i = 2, 1 \leq i \leq n\}$$

Let us calculate the total utilization achieved based on a possible period assignment. For this purpose, we can write

$$U = \sum_{1 \leq i \leq n+2} \frac{C_i}{T_i} = \sum_{i \in J_1} C_i + \sum_{i \in J_2} \frac{C_i}{2} + C_{n+1} + \frac{C_{n+2}}{2}. \tag{8}$$

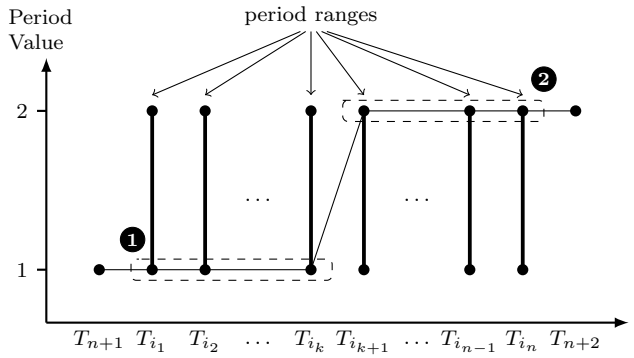

Fig. 1 Partitioning of the task set into two subsets; $K_1 = \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}\} \cup \{\tau_{n+1}\}$ and $K_2 = \{\tau_{i_{k+1}}, \ldots, \tau_{i_n}\} \cup \{\tau_{n+2}\}$. Note that here $(i_1, i_2, \ldots, i_n)$ is a permutation of $\{1, 2, \ldots, n\}$.

Substituting the value of C_is from (6) and (7) yields

$$\begin{aligned} U &= \sum_{i\in J_1} \frac{4s(a_i)}{3S+3} + \sum_{i\in J_2} \frac{1}{2}\frac{4s(a_i)}{3S+3} + \frac{2}{3S+3} + \frac{1}{3S+3} \\ &= \frac{4}{3S+3}\left(\sum_{i\in J_1} s(a_i) + \sum_{i\in J_2} \frac{s(a_i)}{2}\right) + \frac{3}{3S+3} \\ &= \frac{4}{3S+3}\left(\sum_{i\in J_1} \frac{s(a_i)}{2} + \sum_{i\in J_1} \frac{s(a_i)}{2} + \sum_{i\in J_2} \frac{s(a_i)}{2}\right) + \frac{3}{3S+3} \\ &= \frac{4}{3S+3}\left(\sum_{i\in J_1} \frac{s(a_i)}{2} + \frac{S}{2}\right) + \frac{3}{3S+3}. \end{aligned} \tag{9}$$

As a result, $U = 1$ if and only if

$$1 = \frac{4}{3S+3}\left(\sum_{i\in J_1} \frac{s(a_i)}{2} + \frac{S}{2}\right) + \frac{3}{3S+3}, \tag{10}$$

or equivalently,

$$\frac{3S}{4} = \sum_{i\in J_1} \frac{s(a_i)}{2} + \frac{S}{2}, \tag{11}$$

which means

$$\frac{S}{2} = \sum_{i\in J_1} s(a_i). \tag{12}$$

Equation (12) holds if and only if in the PART problem there exists a subset of A whose total size of items equals to $S/2$. This means that the set A can be partitioned into two subsets A_1 and A_2 with $S_1 = S_2$. As a result, the answer to the PART problem is positive if and only if $U = 1$ in the UHPA problem.

Theorem 2 *The UHPA problem is at least weakly NP-hard.*

Proof The proposed transformation method can reduce any instance of the PART problem to an UHPA problem in polynomial time. As a result, if there exists a solution approach to the UHPA problem with a complexity better than a pseudo-polynomial time algorithm, then there exists an algorithm for the PART problem with the same computational complexity as shown in Lemma 1. As a result, the UHPA problem is at least as hard as the PART problem. According to this fact, and also using Theorem 1, it is implied that the UHPA problem is at least weakly NP-hard.

3.3 Complexity of CHPA

In order to show the complexity class of the problem, we provide a polynomial time reduction technique from the PART problem to CHPA. Toward this, consider an arbitrary instance of the PART problem with a set of items $A = \{a_1, a_2, \ldots, a_n\}$. Let S denote the sum of A's items size, i.e., $S = \sum_{i=1}^{n} s(a_i)$. Corresponding to this problem, we construct an instance of CHPA which consists of a set of $n+2$ real-time tasks, denoted as $\{\tau_1, \tau_2, \ldots, \tau_{n+2}\}$. The WCET and weight of task τ_i, for $1 \leq i \leq n$, are assigned as

$$C_i = w_i = \frac{4s(a_i)}{3S+3}. \tag{13}$$

Furthermore, we put $I_i = [1, 2]$, for $1 \leq i \leq n$, as the interval from which τ_i's period has to be selected. For τ_{n+1} and τ_{n+2}, we choose

$$C_{n+1} = w_{n+1} = C_{n+2} = w_{n+2} = \frac{2}{3S+3}, \tag{14}$$

$I_{n+1} = [1, 1]$, and $I_{n+2} = [2, 2]$.

Let $\{T_1, T_2, \ldots, T_{n+2}\}$ denote a possible period assignment. We define J as the weighted sum of these periods, namely $J = \sum_{i=1}^{n+2} w_i T_i$. Also, assume J^* to denote the optimal (i.e., minimum) value of J while the problem constraints are satisfied. We show, for the constructed instance of CHPA, that we have $J^* = 2$ if and only if the instance of the PART problem is a positive one. For this purpose, we use the following lemma.

Lemma 2 *For the instance of CHPA specified above, it holds that*

$$J^* \geq 2. \tag{15}$$

Proof Let T_i denote the period of task τ_i assigned by a solution to CHPA. According to the specified task parameters, the period of τ_{n+1} and τ_{n+2}, namely T_{n+1} and T_{n+2}, are forced to be 1 and 2, respectively. Further, for the period of other tasks, i.e., T_i, for $1 \leq i \leq n$, there are only two options, i.e., 1 and 2 (otherwise, $T_i/T_{n+1} \notin \mathbb{N}$ and $T_{n+1}/T_i \notin \mathbb{N}$, which violates constraint (2c)). Based on this observation, we define

$$\mathbb{C}_1 = \sum_{\substack{1 \leq i \leq n+2 \\ T_i = 1}} C_i,$$

$$\mathbb{C}_2 = \sum_{\substack{1 \leq i \leq n+2 \\ T_i = 2}} C_i.$$

Further, we define $\mathbb{C}$ as the total sum of WCETs, i.e.,

$$\begin{aligned} \mathbb{C} = \sum_{i=1}^{n+2} C_i &= \sum_{i=1}^{n} \frac{4s(a_i)}{3S+3} + \frac{4}{3S+3} \\ &= \frac{4}{3S+3}\left(\sum_{i=1}^{n} s(a_i) + 1\right) = \frac{4}{3S+3}(S+1) = \frac{4}{3}. \end{aligned} \tag{16}$$

From these definitions, it is revealed that $\mathbb{C} = \mathbb{C}_1 + \mathbb{C}_2$. Then, the total utilization achieved based on the assumed period assignment can be written as

$$U = \sum_{i=1}^{n+2} \frac{C_i}{T_i} = \sum_{i:T_i=1} C_i + \sum_{i:T_i=2} \frac{C_i}{2} = \mathbb{C}_1 + \frac{\mathbb{C}_2}{2} = \frac{2\mathbb{C}_1 + \mathbb{C}_2}{2} = \frac{\mathbb{C}_1 + \mathbb{C}}{2} \tag{17}$$

Also, the respective value of J can be computed as

$$J = \sum_{i=1}^{n+2} w_i T_i = \sum_{i:T_i=1} w_i + \sum_{i:T_i=2} 2w_i.$$

Thus, according to (13) and (14), which indicate $w_i = C_i$, for $1 \leq i \leq n+2$, we have

$$J = \mathbb{C}_1 + 2\mathbb{C}_2 = 2\mathbb{C}_1 + 2\mathbb{C}_2 - \mathbb{C}_1 = 2\mathbb{C} - \mathbb{C}_1. \tag{18}$$

From (17) it follows that $\mathbb{C}_1 = 2U - \mathbb{C}$. Replacing $\mathbb{C}_1$ from this relation in (18) yields

$$J = 2\mathbb{C} - (2U - \mathbb{C}) = 3\mathbb{C} - 2U. \tag{19}$$

Finally, from (16), it follows that

$$J = 3(\frac{4}{3}) - 2U = 4 - 2U. \tag{20}$$

Consequently, since $U \leq 1$ (according to (2d)), we have $J \geq 4 - 2 = 2$, which implies $J^* \geq 2$. □

Now we establish a relation between the PART problem and the corresponding instance of CHPA.

Lemma 3 *A given instance of the PART problem is positive if and only if* $J^* = 2$ *in the constructed instance of CHPA.*

Proof We first show that if the given PART problem is a positive instance, then the optimal period assignment satisfies $J^* = 2$. To observe this property, let A_1 and A_2 denote the sets related to a solution of the PART problem. We assign the period of the tasks such that $T_i = 1$ if $a_i \in A_1$ and $T_i = 2$ if $a_i \in A_2$, for $1 \leq i \leq n$. As the total size of the items in A_1 is equal to that of A_2, we will have $\mathbb{C}_1 = \mathbb{C}_2 = \mathbb{C}/2 = 2/3$. Subsequently, from (18), it follows that $J = 2$. According to Lemma 2, this means $J^* = 2$.

Next, we show that if $J^* = 2$, then the given instance of the PART problem is positive. We first notice that, due to (18) and as $\mathbb{C} = 4/3$, the equality $J^* = 2$ implies $\mathbb{C}_1 = 2/3$. Also, since $\mathbb{C}_1 + \mathbb{C}_2 = \mathbb{C}$, we get $\mathbb{C}_2 = 2/3$, too. This means that the set of WCETs can be divided into two sets with equal sum. Referring to (13) and (14) for determining the value of WCETs, it is implied that the set of numbers in the original PART problem is divisible into two sets of which the sum of the numbers are equal. Consequently, the instance of the PART problem is positive, which completes the proof. □

Theorem 3 *CHPA is at least weakly NP-hard.*

Proof Any given instance of the PART problem can be transformed to an instance of CHPA in polynomial time using the presented method. Suppose CHPA to be subject to a solution approach with a complexity better than pseudo-polynomial time. According to Lemma 3, this implies that the original instance of the PART problem can be decided by an algorithm with a complexity better than pseudo-polynomial time, which contradicts with Theorem 1. As a consequence, CHPA is at least weakly NP-hard. □

4 Approximate Solution

In this section, we deal with the problem of optimal harmonic period assignment, in which the goal is to minimize a weighted sum of the periods while we ensure that the periods are harmonic and the utilization is smaller than or equal to 1, referred to as the CHPA$^\infty$ problem. As a reference case we use the optimal solution for the relaxed problem, when the periods are not constrained to be harmonic. We then propose two approximation algorithms for the problem—one with linear and one with quadratic complexity—and show that the error factor of the approximations is bounded by 1.125.

4.1 Optimal Solution for the Relaxed Problem

We start by relaxing the harmonic constraint (namely (2c)) and let the period ratios to be any arbitrary value. As expressed in the following lemma, this relaxed version of the problem can be solved in linear time.

Lemma 4 *Let $(T_1^*, \ldots, T_n^*)$ be the solution of the relaxed problem. In other words,*

$$\sum_{1 \leq i \leq n} w_i T_i^* \leq \sum_{1 \leq i \leq n} w_i T_i, \tag{21}$$

for any period assignment $(T_1, \ldots, T_n)$ which satisfies (2d). Then, T_i^* can be obtained by

$$T_i^* = \sqrt{\frac{C_i}{w_i}} \sum_{1 \leq l \leq n} \sqrt{w_l C_l}. \tag{22}$$

Proof The proof has been previously presented by Cervin et al. (Sect. 3.5 of [23]). It is worth noting that, although in [23] C_is are assumed as *average* execution times, the problem we consider here is mathematicly the same as the one in [23]. As a result, the proof holds for our case as well.

Corollary 1 *Let U^* denote the total utilization of the task set when periods are assigned according to* (22)*. Then, it holds that $U^* = 1$.*

Proof By definition, we have $U^* = \sum_{i=1}^{n} \frac{C_i}{T_i^*}$. Replacing T_i^* from (22) in this relation implies

$$U^* = \sum_{i=1}^{n} \frac{\sqrt{w_i C_i}}{\sum_{l=1}^{n} \sqrt{w_l C_l}} = 1,$$

which completes the proof.

4.2 A Linear-Complexity Approximate Solution

In this section, we present an approximation algorithm for the harmonic case, based on the optimal solution obtained above.

Let $(T_1^*, \ldots, T_n^*)$ be the optimal solution of the relaxed problem indicated in Lemma 4. Further, we define J^* as

$$J^* = \sum_{i=1}^{n} w_i T_i^*. \tag{23}$$

Obviously J^* provides a *lower bound* for the solution of the problem with the harmonic constraint, i.e., (2c). In the following, we first present a fast but non-optimal harmonic period assignment algorithm using periods T_i^*. Then, we calculate a bound for the maximum error of the algorithm. In the subsequent subsections, we assume that tasks have been indexed such that $T_{i-1}^* \leq T_i^*$, $\forall i; 2 \leq i \leq n$.

4.2.1 The Approximation Algorithm

In our algorithm, we first assign $T_1 = T_1^*$. Then, for each T_i, for $i > 1$, we find the smallest *harmonic* period (with respect to the period of the previous task) which is not smaller than T_i^*. For instance, the period of the second task, denoted by T_2, is determined as

$$T_2 = \left\lceil \frac{T_2^*}{T_1} \right\rceil T_1. \tag{24}$$

Similarly, for the i-th task, we will have

$$T_i = \left\lceil \frac{T_i^*}{T_{i-1}} \right\rceil T_{i-1}. \tag{25}$$

From Eq. (25), it is seen that $T_i \geq T_i^*$ for all i; thus, when the periods, i.e., T_is, for $1 \leq i \leq n$, are assigned according to (24) and (25), the total utilization of the system is equal to or less than 1. This is because the period assignment with T_i^* implies a utilization of 1 (based on Corollary 1). As a result, we can scale down each period value such as T_i by a factor of U until the total utilization becomes 1.

The pseudo-code of the algorithm is presented in Algorithm 1. The calculated periods are harmonic because the ratio between every two consecutive period is an integer value. Further, in Lines 11 to 13, the resulting periods are scaled so that the total utilization becomes 1. Consequently, the obtained periods satisfy (2d), and hence, comprise a valid solution to the problem.

Algorithm 1: Simple Period Assignment

input: A WCET C_i and a weight w_i for each task τ_i.
`// Indexing is done such that` $C_i/w_i \leq C_{i+1}/w_{i+1}$`, for` $1 \leq i < n$.
output: A set of harmonic periods T_i, $1 \leq i \leq n$.

```
1  begin
3      σ ← Σ_{1≤l≤n} sqrt(w_l C_l);
4      for i ← 1 to n do
5          T_i^* ← sqrt(C_i/w_i) σ;
6      end
7      T_1 ← T_1^*;
8      for i ← 2 to n do
9          T_i ← ⌈T_i^*/T_{i-1}⌉ T_{i-1};
10     end
       // Scaling step
11     u = Σ_{1≤i≤n} C_i/T_i ;
12     for i ← 1 to n do
13         T_i ← u T_i ;
14     end
15 end
```

4.2.2 A Trivial Error Bound

We first give a trivial error bound by showing that the periods can no more than double when using the approximation algorithm:

Lemma 5 *For any period T_i obtained from Algorithm 1, we have*

$$T_i < 2T_i^* \tag{26}$$

Proof The proof is by induction. For the base case ($i = 1$) we have $T_1 = T_1^* < 2T_1^*$. Also, for $i = 2$, we have

$$\begin{aligned} T_2 &= \left\lceil \frac{T_2^*}{T_1} \right\rceil T_1 \\ &< \left(\frac{T_2^*}{T_1} + 1 \right) T_1 \\ &= T_2^* + T_1 = T_2^* + T_1^* \end{aligned}$$

Since we assumed that the periods (i.e., T_i^*s) are sorted in a non-decreasing order, we have $T_1^* \leq T_2^*$. As a result,

$$\begin{aligned} T_2 &< T_2^* + T_1^* \\ &\leq T_2^* + T_2^* = 2T_2^* \end{aligned}$$

Now, assuming that (26) holds for i, we show that it will hold for $i + 1$ as well, for any $i < n$. We consider two cases for T_{i+1}^* with respect to T_i.

Case 1: $T_{i+1}^* \leq T_i$. In this case, we have $\left\lceil \frac{T_{i+1}^*}{T_i} \right\rceil = 1$; hence, $T_{i+1} = T_i$ (due to (25)). According to the assumption $T_i < 2T_i^*$, and since $T_i^* \leq T_{i+1}^*$, it follows that

$$T_{i+1} = T_i < 2T_i^* \leq 2T_{i+1}^*. \tag{27}$$

Case 2: $T_{i+1}^* > T_i$. In this case, we have

$$\begin{aligned} T_{i+1} &= \left\lceil \frac{T_{i+1}^*}{T_i} \right\rceil T_i \\ &< \left(\frac{T_{i+1}^*}{T_i} + 1 \right) T_i \\ &= T_{i+1}^* + T_i \\ &< T_{i+1}^* + T_{i+1}^* \\ &= 2T_{i+1}^* \end{aligned}$$

which completes the induction.

Corollary 1 *The approximation error with Algorithm 1 is smaller than 2, i.e.,*

$$\frac{J}{J^*} < 2.$$

Proof Since the cost J depends linearly on the periods, it follows from Lemma 5 that the cost will less than double. Finally, the rescaling step can only make the periods smaller and can hence only give a lower cost.

4.3 A Tight Error Bound for Algorithm 1

Now, we derive a tight upper bound on the error of the approximation algorithm.

Theorem 4 *The worst-case relative error of Algorithm 1 is* $\frac{9}{8}$.

Proof Without loss of generality, assume that the weights are scaled so that

$$J^* = \sum_{i=1}^{n} w_i T_i^* = 1.$$

Using (22) we have

$$J^* = \sum_{i=1}^{n} \left(w_i \sqrt{\frac{C_i}{w_i}} \sum_{l=1}^{n} \sqrt{w_l C_l} \right) = \left(\sum_{i=1}^{n} \sqrt{w_i C_i} \right)^2 = 1,$$

which implies that

$$\sum_{i=1}^{n} \sqrt{w_i C_i} = 1$$

and

$$T_i^* = \sqrt{\frac{C_i}{w_i}}.$$

This further implies that

$$U_i^* = \frac{C_i}{T_i^*} = \sqrt{w_i C_i} = w_i T_i^*.$$

From Lemma 5 we know that the periods can no more than double in the harmonization step. We express this as

$$T_i = (1 + \beta_i) T_i^*, \quad 0 \leq \beta_i < 1, \quad i = 2, \ldots, n.$$

The new cost after harmonization is given by

$$\begin{aligned} \hat{J} &= U_1^* + \sum_{i=2}^{n} w_i T_i = U_1^* + \sum_{i=2}^{n} (1 + \beta_i) U_i^* = \sum_{i=1}^{n} U_i^* + \sum_{i=2}^{n} \beta_i U_i^* \\ &= 1 + \sum_{i=2}^{n} \beta_i U_i^*. \end{aligned}$$

Similarly, the utilization after harmonization is calculated as

$$\begin{aligned} \hat{U} &= U_1^* + \sum_{i=2}^{n} \frac{C_i}{(1 + \beta_i) T_i^*} = U_1^* + \sum_{i=2}^{n} \frac{C_i}{T_i^*} \left(1 - \frac{\beta_i}{1 + \beta_i} \right) \\ &= 1 - \sum_{i=2}^{n} \frac{\beta_i}{(1 + \beta_i)} U_i^*. \end{aligned}$$

In the final rescaling step, all periods are multiplied by $\hat{U}$. Since the cost depends linearly on the periods, this means that the final cost after rescaling becomes

$$J = \hat{J}\hat{U} = \left(1 + \sum_{i=2}^{n} \beta_i U_i^*\right)\left(1 - \sum_{i=2}^{n} \frac{\beta_i U_i^*}{1+\beta_i}\right).$$

Using the fact that $\frac{\beta_i}{2} < \frac{\beta_i}{1+\beta_i} \leq \beta_i$ for $0 \leq \beta_i < 1$, we find that

$$J < \left(1 + \sum_{i=2}^{n} \beta_i U_i^*\right)\left(1 - \sum_{i=2}^{n} \frac{\beta_i U_i^*}{2}\right).$$

The right-hand side of the inequality is maximized when $\sum_{i=2}^{n} \beta_i U_i^* = \frac{1}{2}$, yielding the upper bound

$$J < \left(1 + \frac{1}{2}\right)\left(1 - \frac{1}{4}\right) = \frac{9}{8}.$$

The bound becomes arbitrarily tight when $\beta_i \to 1$ and $\sum_{i=2}^{n} U_i^* = \frac{1}{2}$. □

Example 1 Assume two tasks, with execution times $C_1 = C_2 = 0.5$ and weights $w_1 = 0.501$ and $w_2 = 0.499$. The optimal periods are $T_1^* = 0.999$ and $T_2^* = 1.001$ with the optimal cost $J^* = 1.000$. Applying Algorithm 1, we obtain the harmonic periods $T_1 = 0.750$ and $T_2 = 1.500$ with the cost $J = 1.124$.

4.4 A More Effective Approximation Algorithm

The approximation algorithm introduced in Sect. 4.2.1 tries to find the closest harmonic period assignment starting from $T_1 \leftarrow T_1^*$. It means that the solution which is found through Lines 1 to 10 of Algorithm 1 includes the first optimal period. The result of this assignment can be improved if we set the *base* of the search on other T_i^* values as well. For example, if we have $T^* = \{42, 56, 98\}$, the resulting periods from Lines 1 to 10 of Algorithm 1 will be $T = \{42, 84, 168\}$ with total sum of 294 while if we had used $T = \{56, 56, 112\}$, the total sum would be 224. The latter solution can be obtained if we start to assign $T_2 \leftarrow T_2^*$ and then try to find the smallest harmonic values which are larger than the other T_i^*s. For the same reason, any value in T^* can be the *base* candidate and may result in a solution with smaller error than Algorithm 1. This is the basic idea of our second approximation algorithm which is shown in Algorithm 2. This idea has been originally used by Han et al. in [5] for DCT algorithm. However, their goal was to find the largest harmonic period which is smaller than the original given periods of the tasks.

The approximation error of Algorithm 2 will not be larger than that of Algorithm 1 because in the worst-case, Algorithm 2 returns an assignment which starts from $T_1 \leftarrow T_1^*$. In all other cases, the final result of Algorithm 2 can only be better than Algorithm 1 due to the condition in Line 17. Consequently, the

Algorithm 2: DCT-Based Period Assignment

input: A WCET C_i and a weight w_i for each task τ_i.
// Indexing is done such that $C_i/w_i \leq C_{i+1}/w_{i+1}$, $1 \leq i < n$.
output: A set of harmonic periods T_i, $1 \leq i \leq n$.

```
1  begin
2      Obtain T* values from (22);
3      σ^min ← null;
4      for i ← 1 to n do
5          T'_i ← T*_i;
6          for j ← i + 1 to n do
7              T'_j ← ⌈T*_j / T'_{j-1}⌉ T'_{j-1};
8          end
9          for j ← i − 1 down to 1 do
10             T'_j ← T'_{j+1} / ⌊T'_{j+1} / T*_j⌋;
11         end
           // Scaling step
12         u = Σ_{1≤i≤n} C_i / T'_i;
13         for i ← 1 to n do
14             T'_i ← u T'_i;
15         end
16         σ ← Σ_{1≤j≤n} w_j T'_j;
17         if σ < σ^min or σ^min = null then
18             σ^min ← σ;
19             for i ← 1 to n do
20                 T_i ← T'_i;
21             end
22         end
23     end
24 end
```

relative error of Algorithm 2 with respect to the optimal period assignment is upper-bounded by that of Algorithm 1.

According to our experimental results (shown in the next section), both algorithms produce near-optimal results and the difference in their performance is small. However, it is worth noting that their computational complexity is different: Algorithm 2 is $O(n^2)$ while Algorithm 1 is linear-time, i.e., $O(n)$.

5 Experimental Results

In this section, we evaluate the performance and effectiveness of our proposed approximation algorithms presented in Sect. 4. In Sect. 5.1 we measure the efficiency of our approximation algorithms in finding harmonic periods using synthetic task sets while in Sect. 5.2 we measure the effectiveness of our algorithms for a set of benchmark control applications.

5.1 Measuring Efficiency of the Algorithms

In this set of experiments we compare our two approximation algorithms with an optimal solution based on an exhaustive search over all relevant harmonic period assignments. This optimal algorithm returns a harmonic period assignment with utilization 1 which minimizes (2a). This algorithm has been derived from the Forward approach in [10], and it is based on an exhaustive search which iterates over all possible integer multipliers between any two period ranges, and hence, it searches the whole solution space to find an assignment with the minimum error. We have added a pruning rule to limit the results to the assignments with $U \leq 1$. In the experiments we measure the *relative error* of the proposed algorithms as

$$E = \frac{\sum_{i=1}^{n} w_i T_i}{\sum_{i=1}^{n} w_i T_i^*} \tag{28}$$

where T_i^* is obtained from (22). It is worth noting that T_i^* values are not necessarily harmonic and hence, even the optimal algorithm based on the Forward approach may have non-zero error. In the following experiments we evaluate the effect of the WCET, number of tasks, and weights on the relative error of our proposed algorithms.

5.1.1 The Effect of WCET

In the first experiment, we consider the effect of relative ratio of WCET values, i.e., C_i/C_{i-1}. The parameter of experiment is the maximum ratio between any two consecutive WCET values (sorted in an ascending order). For this experiment we assume a task set with 10 tasks (later in Sec. 5.1.2 we will evaluate the effect of different number of tasks). The WCET of each task is selected randomly with uniform distribution so that $C_i \in [C_{i-1}, kC_{i-1}]$ where $C_1 \in [1, 10]$ and k is the maximum WCET ratio. We assume all tasks have the same weight, i.e., $\forall i, w_i = 1$. For each value of k we generate 10,000 random task sets and report the average, maximum, and minimum relative errors that are seen in those task sets.

As it is shown in Fig. 2, when the relative ratio of two consecutive WCET is small, Algorithm 1 has higher error because it only finds the harmonic assignments starting from T_1^*. Since valid assignments must be larger than T_i^* values, if $T_2^*/T_1^* \approx 1$, Algorithm 1 assigns $T_2 \leftarrow 2T_1$ which will be much larger than T_2^*. However, since $2T_1^*$ will be much larger than T_i^* $(2 \leq i \leq n)$, it is a safe assignment for other T_i values. Consequently, the resulting harmonic periods will be $T = \{T_1^*, 2T_1^*, 2T_1^*, \ldots, 2T_1^*\}$ while the period assignment from Algorithm 2 is $T' = \{T_n^*, T_n^*, \ldots, T_n^*\}$. It happens because if we have small ratio between consecutive C_i values, then $T_n^* < 2T_1^*$.

In the next experiment, we evaluate the effect of selection range for C_i values. In this experiment we assume $n = 10$, $\forall i, w_i = 1$, and each C_i is selected randomly with a uniform distribution from range $[1, 10^\sigma]$ where σ

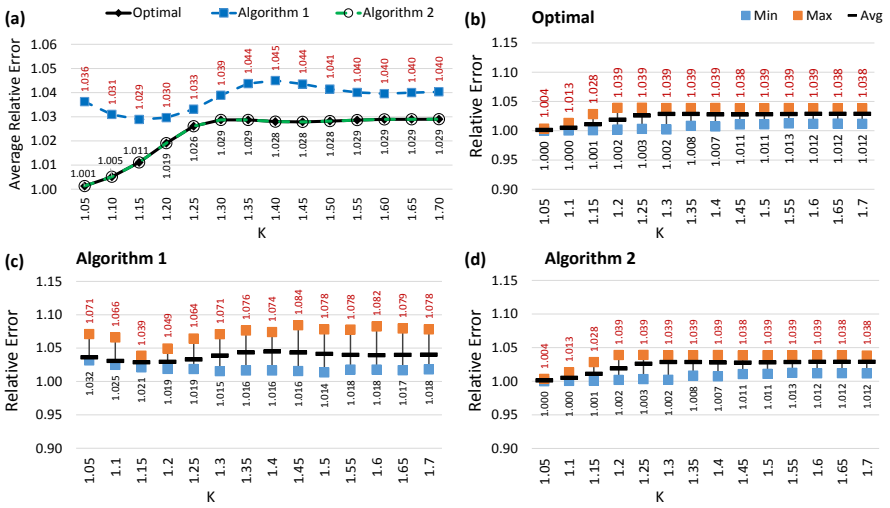

Fig. 2 The relative error of the algorithms. The WCET of each task has been randomly selected so that $C_i \in [C_{i-1}, kC_{i-1}]$ where $C_1 \in [1, 10]$. In diagram (a), the optimal method and Algorithm 2 are overlapped.

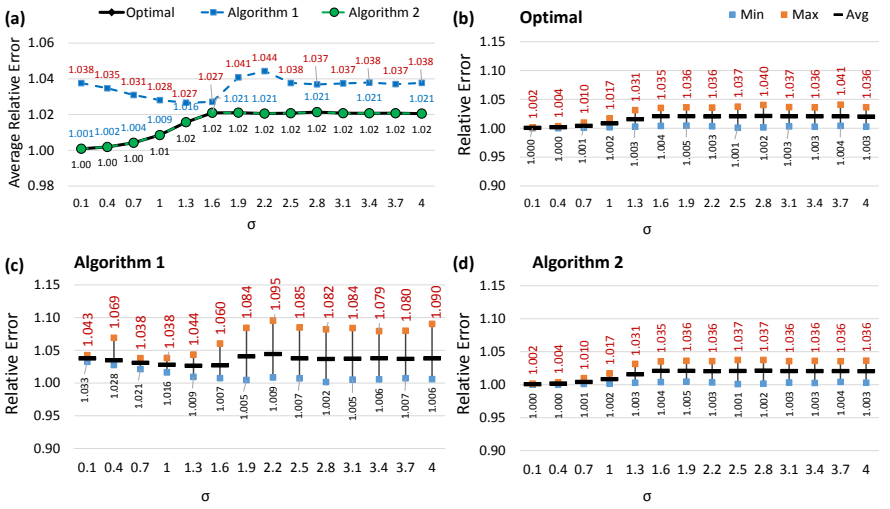

Fig. 3 The relative error of the algorithms. The WCET of each task has been selected from $[1, 10^{\sigma}]$. In diagram (a), the optimal method and Algorithm 2 are overlapped.

is the parameter of the experiment and shows the wideness of the ranges. Fig. 3-(a) to (d) shows the results of this experiment.

When σ is small, e.g., smaller than 1, WCETs are selected from a narrower range which means that they are more similar to each other. As discussed in Sect. 4, when WCETs are similar, Algorithm 1 has a large relative error as it can be seen in Fig. 3-(a) and (c). In this situation, however, both Algorithm 2

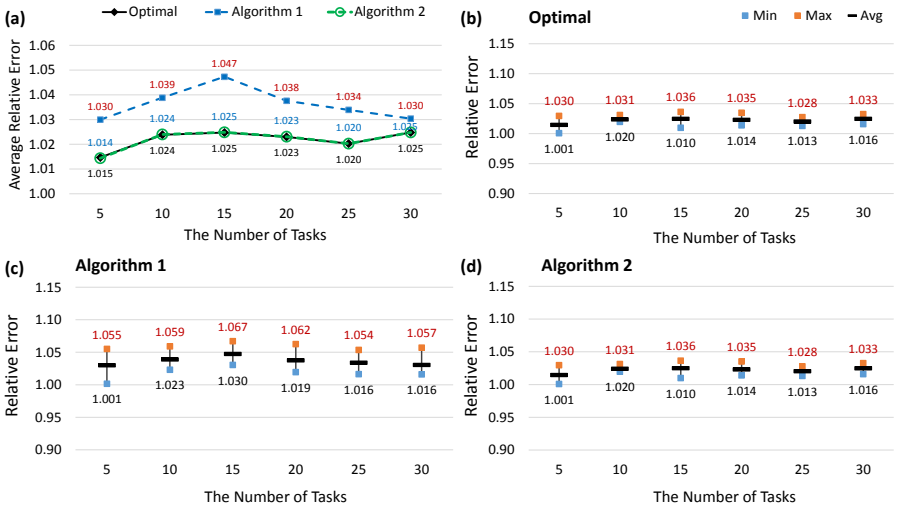

Fig. 4 The relative error of the algorithms. The WCET of each task has been selected from $[1, 500]$. In this diagram, the optimal method and Algorithm 2 are overlapped.

and the optimal algorithm are able to find harmonic periods with small error because since the WCETs are similar, T_i^* values are almost the same which means that the optimal harmonic assignment will be the one that assigns one period to all tasks. Since Algorithm 1 starts by assigning T_1^* as T_1, it may not be able to assign the same period to all other tasks, and hence, it will suffer from larger errors. We will study this particular case later in Sect. 5.1.4.

From $\sigma = 0.1$ to 1.6 we see a decreasing pattern in the relative error of Algorithm 1. The reason is that since we select 10 values from the range $[10, 10^{1.6}]$, the chance that the resulting WCETs have larger relative ratio increases with the increase in σ. Consequently, the relative error of Algorithm 1 decreases since the situation becomes better.

With the increase in σ after $\sigma = 1.6$, WCET values increase as they can be selected from a larger range. This increase in the average values of WCETs affect the result of $\sum_{i=1}^{n} \sqrt{w_i C_i}$ which in turn leads to have large T_i^* (since they are obtained from (22)). In addition, if $\sum_{i=1}^{n} \sqrt{w_i C_i}$ is a large value, the effect of $\sqrt{C_i/w_i}$ on T_i^* decreases, particularly because $\sqrt{x}$ function does not grow as fast as x. Consequently, T_i^* values become large and become relatively close to each other. Due to this effect, the relative error of our algorithms remains almost constant as it can be seen for large values of σ in Fig. 3-(a) and large values of WCET ratio in Fig. 2-(a).

5.1.2 The Effect of the Number of Tasks

Next, we consider the effect of the number of tasks, i.e., n on the relative error of the algorithms. The WCET of each task is selected randomly with a

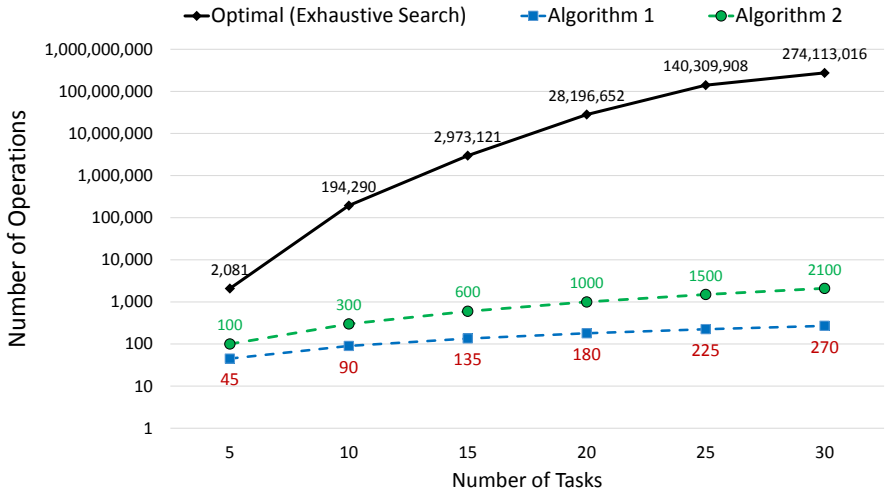

Fig. 5 The number of operations required by different algorithms to find harmonic periods. Note that the vertical axis has a logarithmic scale.

uniform distribution from interval $[1, 500]$ while n varies from 5 to 30. In this experiment we assume that all weights are equal to 1.

Fig. 4-(a) demonstrates the average relative error as a function of the number of tasks. As it is shown, Algorithm 2 still performs as good as the optimal algorithm and is able to find harmonic periods with the minimum error. Fig. 4-(b) to (d) confirm the fact that the algorithms provide predictable assignments where the maximum observed relative error is very close to the average and the minimum observed relative error. It is worth noting that when n increases, the number of values selected from the range increase which in turn increases $\sum_{i=1}^{n} \sqrt{w_i C_i}$. Thus the same effect that happened for the large σ values in Fig. 3 happens here as well.

Since n directly affects the time consumption of our algorithms, for this experiment we also report the number of algorithmic operations that are needed to be performed until harmonic periods are found by the algorithms. Note that we only count the algorithmic steps rather than single instructions because the number of instructions depend on the platform and the programming language. Thus, we count the operations performed in Agorithm 1 and 2. For example, Line 3 or 11 of Algorithm 1 consist of n operations while Line 5 consists of 1 operation. For the case of the exhaustive search, we do it the same way that is done by Nasri et al. in [10]. Fig. 5 shows the number of operations required to find harmonic periods. As it can be seen, the exhaustive solution has an exponential growth when n increases while both of our algorithms have a polynomial growth w.r.t the number of tasks.

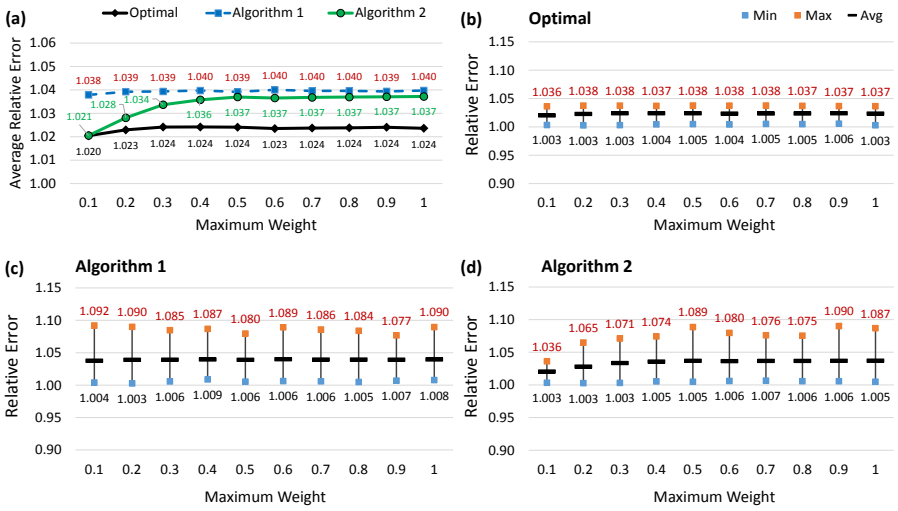

Fig. 6 Relative error of different algorithms as a function of weights. WCETS are selected from $[1, 500]$ and weights are selected from $[0.1, W]$ where W is the horizontal axis of the diagrams.

5.1.3 The Effect of Weights

In the next experiment, we evaluate the effect of weights on the relative error of the algorithms. For this experiment we have 10 tasks and the WCETs are selected with a uniform distribution from range $[1, 500]$. The weight of each task is selected with uniform distribution from $[0.1, W]$ where W shows the maximum weight and is the parameter of the experiment.

Fig. 6-(a) shows the average relative error of the algorithms. When the maximum weight is 0.1, all weights are equal (because the weights will all be 0.1. In that case, Algorithm 2 performs the same as the optimal algorithm. However, by the increase in the length of intervals from which the weights are selected, Algorithm 2 is no longer able to find harmonic periods with the minimum cost because it does not consider the effect of weights when it finds harmonic assignments through Lines 5 to 15. In other words, Algorithm 2 focuses on finding the closest harmonic values and cannot consider the effect of weights. As a result, its decisions will gradually become similar to Algorithm 1 which is ignorant towards the weights too.

Figs. 6-(b) to (d) show the maximum, average, and minimum relative error of the algorithms. As it can be seen, the optimal algorithm is the most predictable since the difference between its maximum and minimum relative errors is about 0.03. This is the case for Algorithm 2 only when all weights are equal to 0.1.

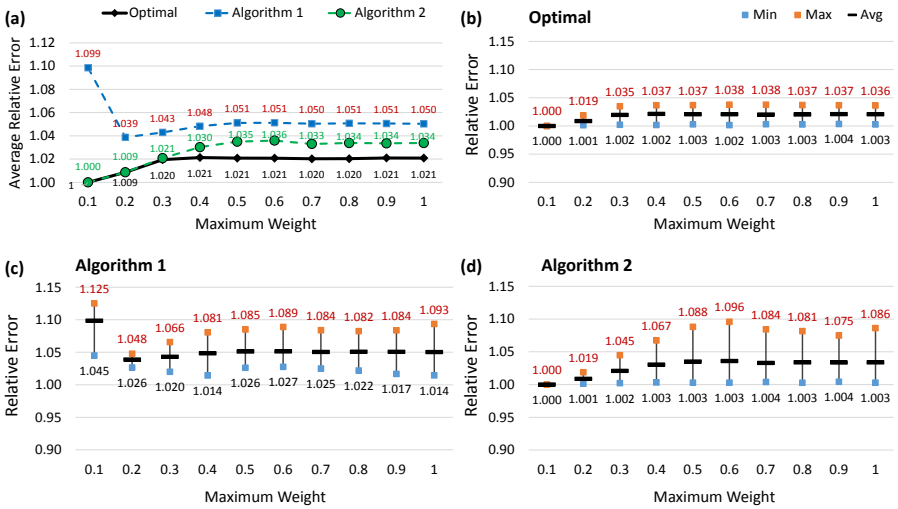

Fig. 7 Relative error of different algorithms as a function of weights for an odd case of WCETs. WCETS are selected from $[10000, 10001]$ and weights are selected from $[0.1, W]$ where W is the horizontal axis of the diagrams.

5.1.4 The Effect of Similar WCET

As it was mentioned in Sect. 4, the maximum error of Algorithm 1 appears when T^* values are almost similar. To study this extreme case we conduct another experiment in which we select the execution times from $[10000, 10001]$ for $n = 10$ tasks. Fig. 7-(a) shows the average relative error of the algorithms as a function of the weights. As it can be seen, for the case where the weights are equal, Algorithm 1 has the largest error. This diagram also shows that the increase in the weights reduces the relative error of Algorithm 1 because they cancel the effect of a harmonic assignment. Moreover, As it can be seen in Fig. 7-(c), the error bound that we have presented in Theorem 4 is tight since there exists task sets whose relative error is $\frac{9}{8} = 1.125$.

Unlike Algorithm 1 that is not efficient when WCETs are similar, Algorithm 2 becomes more efficient because as long as the weights are not different from each other, e.g., for $W = 0.1$ to 0.3, the ratio between the optimal harmonic periods will be 1 (almost in all cases). As a result, the assignment with the smallest error will be the one that assigns the largest value of T_i^* to all tasks. However, when the weights increase, this assignment will not be the best one anymore, and hence, Algorithm 2's decisions deviate from the optimal one.

5.2 Measuring the Effectiveness of the Algorithms for Control Systems

In this subsection, the two algorithms for period harmonization are evaluated in control system examples. To allow for a large number of cases to be investigated, the Jitterbug MATLAB toolbox [27] is used throughout to design the

controllers and to evaluate the resulting performance. We first study a simple example in detail and then present results for 100 sets of randomly generated plants.

5.2.1 A Simple Codesign Example

As a simple codesign example, we assume that three linear plants,

$$\begin{aligned} P_1(s) &= \frac{2}{s^2-1}, \\ P_2(s) &= \frac{2}{s^2}, \\ P_3(s) &= \frac{1}{s(s+1)}, \end{aligned} \tag{29}$$

should be controlled by three control tasks. For each controller, the goal is to minimize the quadratic cost function

$$J_i = \lim_{T\to\infty} \frac{1}{T} \mathrm{E} \int_0^T \left(y_i^2(t) + u_i^2(t) \right) dt, \tag{30}$$

when the plant is subject to white input noise with intensity 1 and white measurement noise with intensity 0.1. Prior to implementation, a continuous-time linear-quadratic-Gaussian (LQG) controller $K_i(s)$ has been designed for each plant to give satisfactory performance. The cost functions are normalized so that $J_i^{\mathrm{ct}} = 1$ corresponds to continuous control. Running all continuous controllers hence gives the ideal overall cost $J^{\mathrm{ct}} = \sum_{i=1}^{3} J_i^{\mathrm{ct}} = 3$.

The execution times are given by $C_1 = 0.144$, $C_2 = 0.175$, $C_3 = 0.102$. These values also represent the minimum possible periods and control delays for the control loops. The weights w_i are found by evaluating the sensitivity towards an increased period for each control loop:

$$w_i = \frac{\partial J_i}{\partial T_i}(C_i). \tag{31}$$

The analysis yields the weights $w_1 = 2.47$, $w_2 = 1.45$, $w_3 = 0.409$. It is seen that the two first plants, which are open-loop unstable, are more sensitive towards a period extension than the last plant. Using Lemma 4, the initial periods are then given by $T_1^* = 0.319$, $T_2^* = 0.458$, $T_3^* = 0.658$. The respective jitter margins for these periods, assuming a constant delay of C_i, are given by $J_{m1} = 0.248$, $J_{m2} = 0.358$, $J_{m3} = 1.40$.

As a baseline, we evaluate two different design methods for the set of sampling periods given by $T^{\mathrm{init}} = T^*/0.99$. (A target utilization of $U = 0.99$ is used for the non-harmonic periods to avoid numerical problems in the evaluation.)

– **Initial LQG design.** Each controller is designed without regard for the scheduling, assuming a constant control delay of C_i.

Table 1 Results for the simple example.

	T_1	T_2	T_3	Constant exec. times	Random exec. times
Initial (J^{init})	0.322	0.463	0.665	5.57	4.43
Delay-aware (J^{da})	0.322	0.463	0.665	5.13	4.55
Algorithm 1, no offset (J^{h1})	0.260	0.520	1.04	5.63	5.38
Algorithm 2, no offset (J^{h2})	0.286	0.571	0.571	5.27	4.69
Algorithm 1, with offset (J^{h1o})	0.260	0.520	1.04	4.86	4.44
Algorithm 2, with offset (J^{h2o})	0.286	0.571	0.571	4.58	4.09

– **Delay-aware LQG design.** The delay of each controller is assumed constant and given by the approximate response-time formula from [22]. Based on this, a standard LQG controller compensating for the fixed delay is designed.

We then evaluate two different methods, where the controllers are redesigned after harmonization:

– **Harmonic LQG design.** Using Algorithm 1 and 2, two sets of harmonic periods are calculated. For each resulting task set, we assign task release offsets to minimize the control delay and design a standard LQG controller for the remaining constant delay. These two sets of controllers are evaluated with or without offset.

We evaluate the cost with constant execution time C, or random execution time in unif$(0.5C, C)$. All the costs are normalized to the continuous-time LQG cost. The resulting periods and the total LQG cost are shown in Table 1, where all numbers have been rounded to three significant digits. It is seen that, in the constant execution time case, the harmonic LQG costs without offset are slightly worse than the initial and delay-aware LQG costs, because the harmonization forces the periods to deviate from their optimal values. The harmonic LQG costs with offset are however lower than the harmonic LQG cost without offset. The reason is that, by using harmonic periods and task offsets, the schedule will give rise to both constant and short delays. This positive effect dominates over the negative effect of having to deviate quite far from the optimal, real-valued periods.

The execution time is rarely constant in real-time systems. It is seen that also for the random execution times, Algorithm 2 with release offsets produces the smallest overall cost.

In the next section, a larger evaluation is performed using randomly generated plant dynamics, showing the variability and confidence of the results obtained.

5.2.2 Randomly Generated Examples

To see whether the results for the simple example above hold in more general cases, sets of three plants have been randomly generated for evaluation from the following four plant families:

– Family A: All plants have two real or complex stable poles. They are drawn from

$$P_A(s) = \frac{K}{\tau^2 s^2 + 2\zeta\tau s + 1}, \tag{32}$$

where $\tau = 10^{-2\alpha}$, $K = 2 \times 10^{\beta}$, $\zeta = 2\gamma$, and α, β, $\gamma \in \text{unif}(0, 1)$.
– Family B: All plants have two real or complex stable poles and one integrator. They are drawn from

$$P_B(s) = \frac{K}{\tau s\,(\tau^2 s^2 + 2\zeta\tau s + 1)}, \tag{33}$$

where $\tau = 10^{-2\alpha}$, $K = 0.3 \times 10^{\beta}$, $\zeta = 2\gamma$, and α, β, $\gamma \in \text{unif}(0, 1)$.
– Family C: All plants have two real or complex stable poles, one stable or unstable zero and one integrator. They are drawn from

$$P_C(s) = \frac{K\,[1 - \text{sgn}(\lambda - 0.5)\tau T_z s]}{\tau s\,(\tau^2 s^2 + 2\zeta\tau s + 1)}, \tag{34}$$

where $\tau = 10^{-2\alpha}$, $K = 1.3 \times 10^{\beta}$, $\zeta = 2\gamma$, $T_z = 10^{\eta-2}$, and α, β, γ, η, $\lambda \in \text{unif}(0, 1)$.
– Family D: All plants have two real or complex stable poles, one stable or unstable pole and one integrator. They are drawn from

$$P_D(s) = \frac{K}{\tau s\,(\tau^2 s^2 + 2\zeta\tau s + 1)\,[1 - \text{sgn}(\lambda - 0.5)\tau T_p s]}, \tag{35}$$

where $\tau = 10^{-2\alpha}$, $K = 10^{\beta}$, $\zeta = 2\gamma$, $T_p = 10^{\eta+1}$, and α, β, γ, η, $\lambda \in \text{unif}(0, 1)$.

The plant parameters have been chosen to give reasonable robustness when combined with the LQG design weights. Also a random time constant τ has been included to generate plants with possibly very different time scales.

25 sets of three plants are randomly generated for each family, and the cost functions are normalized so that 1 corresponds to continuous-time control. The initial sampling periods are again calculated using evaluation of the sensitivity towards an increase in period, Eq. (31). The jitter margin was calculated for all 300 controllers, and in no case was it smaller than 67% of the initial period, indicating that all controllers were reasonably robust.

Using this setup, we evaluate the following costs. In the initial and delay-aware LQG evaluations, we use the utilization target $U = 0.99$, while the harmonic cases use full utilization.

– Initial LQG cost J^{init}. The LQG controllers are designed for T_i^{init} as the period and C_i as the constant delay. For each task, we calculate the first 100 response times to estimate the response time distribution. Using this distribution as the delay distribution in Jitterbug, the LQG cost J_i^{init} is evaluated.

Table 2 Average costs under constant execution times.

	Family A	Family B	Family C	Family D
J^{init}	4.83	4.69	4.38	4.36
J^{da}	4.70	4.53	4.22	4.17
J^{h1}	4.89	4.67	4.36	4.37
J^{h2}	4.75	4.53	4.25	4.21
J^{h1o}	4.60	4.31	4.05	4.05
J^{h2o}	4.45	4.20	3.99	3.92

Table 3 Average costs under random execution times $[0.5C, C]$.

	Family A	Family B	Family C	Family D
J^{init}	4.29	4.07	3.82	3.79
J^{da}	4.33	4.12	3.87	3.84
J^{h1}	4.54	4.30	4.05	4.11
J^{h2}	4.42	4.23	3.97	3.93
J^{h1o}	4.23	3.96	3.74	3.75
J^{h2o}	4.13	3.91	3.71	3.66

- Delay-aware LQG cost J^{da}. We again design the LQG controller with T_i^{init} as the period but now with the approximate delay from [22]. The first 100 response times are used to calculate the response time distribution. Then the distribution is used in the Jitterbug evaluation.
- Harmonic LQG cost J^{h1}, J^{h2}, J^{h1o}, and J^{h2o}. We use T_i^{init} as the initial period and Algorithm 1 and 2 to evaluate the two harmonic task period sets with full utilization. For the no offset case, the LQG controller is designed using a constant delay, equal to the the response time. For the offset case, the LQG controller is designed using a constant delay, equal to the difference between the response time and the start latency. The LQG cost is evaluated with an offset which is equal to the start latency. J^{h1} stands for the LQG cost using Algorithm 1 with no offset, and J^{h1o} stands for the LQG cost using Algorithm 1 with offset. J^{h2} and J^{h2o} are defined similarly.

We consider two options for the execution time: a constant one equal to C and a random one which is uniformly distributed in $(0.5C, C)$. The costs, averaged over 25 generated plant sets for each family, are shown in Tables 2 and 3. As seen from the results, the initial LQG costs are slightly better than the costs of harmonic LQG without offset, and worse than the harmonic LQG with offset. The costs for the harmonization of Algorithm 2 is smaller than those of Algorithm 1, while the computation of Algorithm 2 is more time-consuming.

To see the variability of the results, the LQG costs over all four plant families under constant execution times are shown in the box plot of Fig. 8. The costs are all normalized relative to J^{init}. All the values of J^{da} are smaller than J^{init}, because the delay-aware LQG takes the approximation of the response time, instead of the execution time, as delay into the control design. The values of J^{h1} can be worse or better than J^{init}, because the sampling periods from

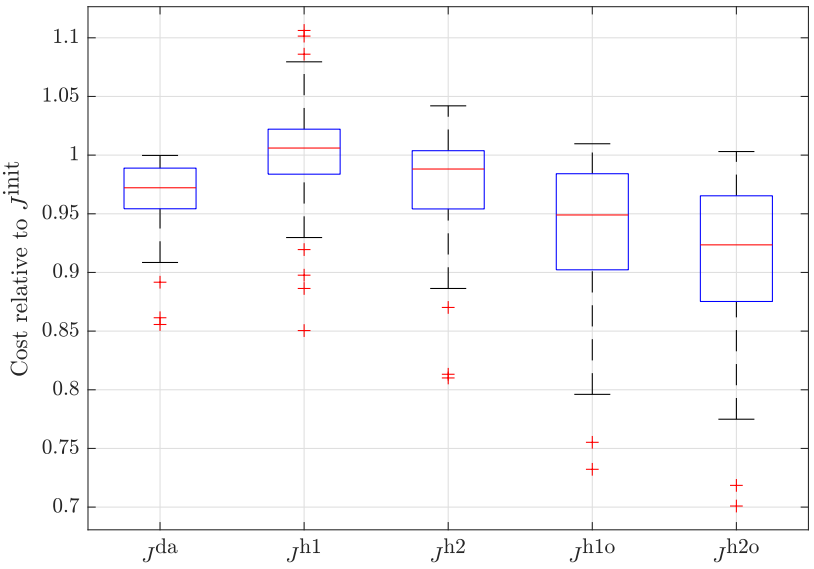

Fig. 8 Costs relative to J^{init} for the 100 randomly generated examples under constant execution times.

Table 4 The complexity of different harmonic period assignment problems.

	$\max U$	$\min \sum w_i T_i$
restricted range	NP-hard in the weak sense (at least)	NP-hard in the weak sense (at least)
unrestricted range	There exists poly. time algorithm	? (There exists poly. time approximation)

Algorithm 1 are different from T_i^*s in Lemma 4, but harmonic. The costs of J^{h2} are smaller than J^{h1}, because the sampling periods calculated by using Algorithm 2 give better overall control performance. Both J^{h1o} and J^{h2o} are smaller than J^{init} on average. The reason is that the offset is adopted in the scheduling, which leads to shorter delay. Furthermore, in almost all cases, J^{h2o} is smaller than J^{init}. The conclusion is that harmonic periods from Algorithm 2 together with release offsets should be the recommended design method for control tasks.

6 Discussions

Table 4 summarizes the known results for the complexity of the optimal harmonic period assignment problem. The results are shown for two different optimization objectives and constrains.

As shown in Sect. 3, when the goal is to maximize U or to minimize the weighted sum of the periods, and we have restricted period ranges (i.e., UHPA and CHPA problems), the harmonic period assignment problem is weakly NP-hard. If periods are not confined by a minimum and maximum value, i.e., they can be any value, then one possible harmonic assignment is

$$T_1 = T_2 = \ldots = T_n = \sum_{i=1}^{n} C_i \tag{36}$$

In (36), period ratio between any two consecutive periods is 1 while the utilization is 1. Hence it is a linear-time solution for UHPA when periods are not restricted. Through Sect. 4 we have shown that there exist polynomial-time approximation algorithms when the goal is to minimize the weighted sum of T_i. However, to the best of the authors' knowledge, there is no known result for the class of hardness of the problem. The latter problem has wide applications in control systems as it can increase their quality of control through minimizing jitters of sampling and actuation.

An interesting observation in our experiments is that, when all tasks incorporate the same weight, Algorithm 2 is as good as the optimal solution based on an exhaustive search. It remains as an open question whether the DCT-based algorithm is really an optimal solution to assign harmonic periods in this situation or not. If it is the case, then CHPA$^{\infty}$ can be solved in polynomial-time using Algorithm 2 for the special case of equal weights.

7 Conclusion

In this paper, we have discussed the hardness of the harmonic period assignment problem in cases where periods must be selected from a given range and the goal is to find a harmonic assignment which maximizes the utilization. We have shown that this problem is weakly NP-hard. It was also shown that the same result holds when the objective is to minimize the weighted sum of periods. We have also considered the problem of minimizing the weighted sum of harmonic periods where there is no lower and upper bound on the valid periods. While the computational complexity of this version of the problem is unknown, we presented two polynomial-time approximation algorithms for it. We have shown that the upper bound of the error of these algorithms with respect to the results of an optimal period assignment algorithm is upper bounded by 1.125. Our algorithms can be used to increase the quality of control in control systems. We have evaluated the proposed algorithms using synthetic task sets as well as benchmark control applications. The results have shown that even though the guaranteed bounded error is 1.125, the second algorithm is as good as the optimal solution based on an exhaustive search when all tasks are assigned the same weight.

As our future work, we intend to investigate the optimality of our second algorithm in the mentioned special case. Moreover, we try to provide efficient

solutions, either using approximation algorithms or heuristics to solve the first problem where periods are bounded to a specified set of intervals. A solution to this problem can be further used in the design space exploration in order to simplify parameter assignment phase. Also we plan to provide a new method to obtain the space of feasible periods for RM.

References

1. D. Seto, J. P. Lehoczky, and Liu Sha. Task period selection and schedulability in real-time systems. In *Real-Time Systems Symposium*, pages 188-198, 1998.
2. E. Bini, M. Di Natale, and G. Buttazzo. Sensitivity analysis for fixed-priority real-time systems. *Real-Time Systems*, 39(1):5-30, 2008.
3. E. Bini and M. Di Natale. Optimal task rate selection in fixed priority systems. In *Real-Time Systems Symposium*, pages 399-409, 2005.
4. Y. Wu, G. Buttazzo, E. Bini, and A. Cervin. Parameter selection for real-time controllers in resource-constrained systems. *IEEE Transactions on Industrial Informatics*, 6(4):610-620, 2010.
5. C. C. Han and H. Y. Tyan. A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms. In *Real-Time Systems Symposium*, pages 36-45, 1997.
6. V. Bonifaci, A. Marchetti-Spaccamela, N. Megow, and A. Wiese. Polynomial-time exact schedulability tests for harmonic real-time tasks. In *Real-Time Systems Symposium*, pages 236-245, 2013.
7. F. Eisenbrand and T. Rothvoß. Static-priority real-time scheduling: response time computation is NP-hard. In *Real-Time Systems Symposium*, pages 397-406, 2008.
8. P Ekberg and W Yi. Uniprocessor feasibility of sporadic tasks with constrained deadlines is strongly coNP-complete. In *Euromicro Conference on Real-Time Systems*, pages 281-286, 2015.
9. C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46-61, 1973.
10. M. Nasri, G. Fohler, and M. Kargahi. A framework to construct customized harmonic periods for real-time systems. In *Euromicro Conference on Real-Time Systems*, pages 211-220, 2014.
11. M. Nasri and G. Fohler. An efficient method for assigning harmonic periods to hard real-time tasks with period ranges. In *Euromicro Conference on Real-Time Systems*, pages 149-159, 2015.
12. M. Nasri, S. Baruah, G. Fohler, and M. Kargahi. On the optimality of RM and EDF for non-preemptive real-time harmonic tasks. *Real-Time Networks and Systems*, pages 331-340, 2014.
13. L. P. Gent and T. Walsh. Analysis of heuristics for number partitioning. *Computational Intelligence*, 14(3):430-451, 1998.
14. C.-S. Shih, S. Gopalakrishnan, P. Ganti, M. Caccamo, and L. Sha. Scheduling real-time dwells using tasks with synthetic periods. In *IEEE Real-Time Systems Symposium*, pages 210–219, 2003.
15. H. Li, J. Sweeney, K. Ramamritham, R. Grupen, and P. Shenoy. Real-time support for mobile robotics. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 10–18, 2003.
16. J.F. Ruiz, GNAT Pro for On-board Mission-Critical Space Applications, In *Ada-Europe International Conference on Reliable Software Technologies*, pages 248–259, 2005.
17. T. Taira, N. Kamata, and N. Yamasaki. Design and implementation of reconfigurable modular humanoid robot architecture. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3566–3571, 2005.
18. I. Mizuuchi, Y. Nakanishi, Y. Sodeyama, Y. Namiki, T. Nishino, N. Muramatsu, J. Urata, K. Hongo, T. Yoshikai, and M. Inaba. An advanced musculoskeletal humanoid kojiro. In *IEEE-RAS International Conference on Humanoid Robots*, pages 294–299, 2007.

19. J. V. Busquets-Mataix, J. J. Serrano, R. Ors, P. Gil, and A. Wellings. Using harmonic task-sets to increase the schedulable utilization of cache-based preemptive real-time systems. In *Workshop on Real-Time Computing Systems and Applications*, pages 195–202, 1996.
20. Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. D. Koutsoukos, and H. Wang. Feedback thermal control for real-time systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 111–120, 2010.
21. S. Kwak and J.-M. Yang. Optimal checkpoint placement on real-time tasks with harmonic periods. *Journal of Computer Science and Technology*, 27(1):105–112, 2012.
22. E. Bini and A. Cervin. Delay-aware period assignment in control systems. In *IEEE Real-Time Systems Symposium*, pages 291–300, 2008.
23. A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén. Feedback-feedforward scheduling of control tasks. *Real-Time Systems*, 23(1):25–53, 2002.
24. D. Seto, J. P. Lehoczky, L. Sha, K. G. Shin. On task schedulability in real-time control systems. In *IEEE Real-Time Systems Symposium*, pages 13–21, 1996.
25. J. Eker, P. Hagander, and K.-E. Årzén. A feedback scheduler for real-time controller tasks. *Control Engineering Practice*, 8(12):1369–1378, 2000.
26. D. Henriksson and A. Cervin. Optimal on-line sampling period assignment for real-time control tasks based on plant state information. In *IEEE Conference on Decision and Control*, pages 4469–4474, 2005.
27. A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén. How does control timing affect performance?. *IEEE Control Systems Magazine*, 23(3):16–30, 2003.
28. Y. Xu, K.-E. Årzén, E. Bini, and A. Cervin. Response time driven design of control systems. In *19th IFAC World Congress*, pages 6098-6104, 2014.
29. Y. Xu, A. Cervin, K.-E. Årzén. Harmonic scheduling and control co-design. In *IEEE Conference on Embedded and Real-Time Computing Systems and Applications*, pages 182–187, 2016.
30. M. Mohaqeqi, M. Nasri, Y. Xu, A. Cervin, K.-E. Årzén. On the problem of finding optimal harmonic periods. In *International Conference on Real-Time Networks and Systems*, pages 171–180, 2016.
31. A. Easwaran, I. Lee, O. Sokolsky, S. Vestal. A compositional scheduling framework for digital avionics systems. In *IEEE Conference on Embedded and Real-Time Computing Systems and Applications*, pages 371–380, 2009.