On the Problem of Finding Optimal Harmonic Periods

Morteza Mohaqeqi(1), Mitra Nasri(2), Yang Xu(3), Anton Cervin(3), Karl-Erik Årzén(3)

(1) Department of Information Technology, Uppsala University, Sweden.
(2) Max Planck Institute for Software Systems (MPI-SWS), Germany.
(3) Department of Automatic Control, Lund University, Sweden.
morteza.mohaqeqi@it.uu.se, mitra@mpi-sws.org,
{yang, anton, karlerik}@control.lth.se

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. This property decreases the jitters which 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. First, we assume that an interval is determined a priori for each task from which its period can be selected. The goal is to assign a (harmonic) period to each task such that the total system utilization is maximized while the task set remains feasible. We show that this problem is (at least) weakly NP-hard. This is shown by reducing the NP-complete number partitioning problem to the mentioned harmonic period assignment problem. Afterwards, we consider a variant of the problem in which the periods are not restricted to a special interval and the objective is to minimize the total weighted sum of the periods with the same feasibility constraint. We present two approximation algorithms for the second problem and show that the maximum error of these algorithms is bounded by a factor of 2. Our evaluations show that, on the average, results of the approximation algorithms are very close to an optimal solution.

1. INTRODUCTION

Selecting appropriate timing parameters for the tasks such that performance objectives are satisfied while 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 which 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 in contrary to the complexity of the problem in the general case which is intractable 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 arbitrary periodic task sets, the respective
We show that the problem is NP-hard by transforming the well-known number partitioning problem to the harmonic period assignment problem. Additionally, we present two polynomial time approximation algorithms for a variant of the problem in which the goal is to minimize the total sum of the periods. We show that the maximum error of our algorithms is upper bounded by a factor of two with respect to an optimal algorithm. Simulation results show that, on average, this error is much smaller than 2, which means that the approximation algorithm behaves very close to the optimal one.

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. Moreover, in these systems, jitters in the sampling and actuation can adversely affect the quality of control (QoC) [21]. 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.

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

2. NOTATIONS AND DEFINITIONS

We consider a set of $n$ real-time tasks $\tau_1, \ldots, \tau_n$ with the set of WCETs $c = \{C_1, C_2, \ldots, C_n\}$. Further, for each task $\tau_i$, an interval $I_i = [T_i^*, T_i]$ is given as the period range. The period of each task $\tau_i$, which is denoted by $T_i$, must be selected from $I_i$. The utilization of $\tau_i$ 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 [21]. 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

\[
\text{maximize } U = \sum_{1 \leq i \leq n} U_i
\]

subject to:

\[
T_i \in I_i, \quad 1 \leq i \leq n \quad (1a)
\]

\[
\frac{T_i}{T_j} \in \mathbb{N} \text{ or } \frac{T_j}{T_i} \in \mathbb{N}, \quad 1 \leq i, j \leq n \quad (1c)
\]

\[
U \leq 1. \quad (1d)
\]

In the subsequent sections, we refer to this problem as the harmonic period assignment (HPA) 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 $U \leq 1$. 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. In this problem, we consider unrestricted periods, meaning $I_i = [0, \infty)$, for $1 \leq i \leq n$.

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

\[
\text{minimize } \sum_{1 \leq i \leq n} w_i T_i
\]

subject to:

\[
\frac{T_i}{T_j} \in \mathbb{N} \text{ or } \frac{T_j}{T_i} \in \mathbb{N}, \quad 1 \leq i, j \leq n \quad (2b)
\]

\[
U \leq 1, \quad (2c)
\]

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

Problem 2 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, e.g., [22].

In [22] 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 the described problems, jitters are forced to be 0 by forcing the periods to be harmonic (assuming constant execution times). Thus, it both allows more accurate description of the controller cost function and less jitter.

In this paper, we propose two approximation algorithms for Problem 2. However, we do not investigate the complexity of this problem, neither for unrestricted periods nor for the case of restricted period ranges. We discuss some related open problems in more details in Section 6.

3. COMPLEXITY RESULT

In this section, we investigate the computational complexity of the HPA problem (i.e., Problem 1). In order to show
the hardness of this problem, we present a polynomial time algorithm for reducing any given instance of the number partitioning (PART) problem to an HPA problem. We first review the PART problem. Then, the transformation method 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 multiset of \( n \) positive integers. The problem is to determine whether \( A \) can be partitioned into two sets \( A_1 \) and \( A_2 \) such that the sum of the numbers in \( A_1 \) equals that of \( A_2 \). More formally, let \( S, S_1, \) and \( S_2 \) denote the sum of the numbers in \( A, A_1, \) and \( A_2 \), respectively. That is,

\[
S = \sum_{a_i \in A} a_i \tag{3}
\]

\[
S_1 = \sum_{a_i \in A_1} a_i \tag{4}
\]

\[
S_2 = \sum_{a_i \in A_2} a_i \tag{5}
\]

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 \( S_1 = S_2 \).

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

### 3.2 Transformation Technique

In this section, a polynomial time method for transforming any given instance of the PART problem to an instance of HPA. 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 HPA 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 HPA problem is specified by a set of \( n + 1 \) real-time tasks. The WCET of \( \tau_i \), for \( 1 \leq i \leq n \), is determined as

\[
C_i = \frac{4a_i}{3S} \tag{6}
\]

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

\[
C_{n+1} = 0, \tag{7}
\]

and \( I_{n+1} = 1 \).

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

\(^1\)A problem is weakly NP-complete if it is NP-complete and it has a pseudo-polynomial time solution.

**Proof.** According to the specified HPA problem, the period of \( \tau_{n+1} \) is forced to be 1. Further, there are only two options for the period of the other tasks, 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. Let \( T_i \) be the period of task \( \tau_i \) assigned by a solution to the HPA problem. We define \( J_1 \) and \( J_2 \) as

\[
J_1 = \{ i \mid T_i = 1 \}
\]

\[
J_2 = \{ i \mid T_i = 2 \}
\]

Let calculate the total utilization achieved based on this period assignment. For this purpose, we can write\(^2\)

\[
U = \sum_{1 \leq i \leq n} C_i/T_i = \sum_{i \in J_1} C_i + \sum_{i \in J_2} C_i/2 \tag{8}
\]

Substituting \( C_i \) by \( 4a_i/3S \) (from (6)) yields

\[
U = \left( \sum_{i \in J_1} \frac{4a_i}{3S} + \sum_{i \in J_2} \frac{1}{2} \frac{4a_i}{3S} \right) \]

\[
= \frac{4}{3S} \left( \sum_{i \in J_1} a_i + \sum_{i \in J_2} a_i/2 \right) \]

\[
= \frac{4}{3S} \left( \sum_{i \in J_1} a_i/2 + \sum_{i \in J_2} a_i/2 + \sum_{i \in J_2} a_i/2 \right) \]

\[
= \frac{4}{3S} \left( \sum_{i \in J_1} a_i/2 + S/2 \right) \tag{9}
\]

As a result, \( U = 1 \) if and only if

\[
1 = \frac{4}{3S} \sum_{i \in J_1} a_i/2 + S/2, \tag{10}
\]

or equivalently,

\[
\frac{3S}{4} = \sum_{i \in J_1} a_i/2 + S/2, \tag{11}
\]

which implies

\[
\frac{S}{2} = \sum_{i \in J_1} a_i. \tag{12}
\]

Equation (12) holds if and only if in the PART problem there exists a subset of \( A \) of which the sum of the numbers 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 HPA problem.

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

**Proof.** The proposed transformation method can reduce any instance of the PART problem to an HPA problem in polynomial time. As a result, if there exists a solution approach to the HPA 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 task \( \tau_{n+1} \) is excluded from the computation since its utilization is zero.
Tasks corresponding to $A_1$

Tasks corresponding to $A_2$

Figure 1: Partitioning of the task set into two subsets; $K_1 = \{\tau_1, \tau_2, \ldots, \tau_k\} \cup \{\tau_{n+1}\}$ and $K_2 = \{\tau_{k+1}, \ldots, \tau_n\}$. Note that here $(i_1, i_2, \ldots, i_n)$ is a permutation of $\{1, 2, \ldots, n\}$.

HPA is at least as hard as the PART problem. According to this fact, and also using Theorem 1, it is concluded that the HPA problem is at least weakly NP-hard.

4. APPROXIMATE SOLUTION

In this section, we consider the problem of optimal harmonic period assignment in which the weighted sum of the periods should be minimized while we ensure that the periods are harmonic and the utilization is smaller than or equal to 1, as specified in Problem 2. We propose two approximation algorithms for the problem. Towards this, we first show how optimal periods can be calculated in polynomial time when the periods are not constrained to be harmonic.

4.1 Optimal Solution for the Relaxed Problem

In this section, we relax the harmonic constraint (namely (2b)) and let the period ratios to be any arbitrary value. As expressed in the following lemma, this relaxed problem can be solved in polynomial time.

**Lemma 2.** Let $(T_1^*, \ldots, T_n^*)$ be the solution of the relaxed problem. In other words,

$$\sum_{1 \leq i \leq n} w_i T_i^* \leq \sum_{1 \leq i \leq n} w_i T_i, \quad (13)$$

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

$$T_i^* = \sqrt{\frac{C_i}{w_i}} \sum_{1 \leq l \leq n} \sqrt{w_l C_l}. \quad (14)$$

**Proof.** The proof has been previously presented in [22] (Section 3.5).

**Corollary 1.** Let $U^*$ denote the total utilization of the task set when periods are assigned according to (14). 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 (14) in this relation implies

$$U^* = \sum_{i=1}^n \frac{\sqrt{w_i C_i}}{\sum_{i=1}^n \sqrt{w_i C_i}} = \frac{1}{\sum_{i=1}^n \sqrt{w_i C_i}} \sum_{i=1}^n w_i C_i = 1.$$  

which completes the proof.

4.2 An Approximate Solution

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

Let $(T_1^*, \ldots, T_n^*)$ be the optimal solution of the relaxed problem indicated in Lemma 2. Further, we define $S^*$ as

$$S^* = \sum_{1 \leq i \leq n} w_i T_i^* \quad (15)$$

Obviously $S^*$ provides a lower bound for the solution of the problem with the harmonic constraint, i.e., (2b). In the following, we first present a fast but non-optimal harmonic period assignment algorithm using periods $T_i^*$. Then, we calculate an upper bound for the maximum error of the algorithm. For this purpose, we show that sum of the periods determined by the approximate solution is no larger than $2S^*$. In the subsequent subsections, we assume that tasks have been indexed such that $T_i^* \leq T_i^* \leq 2T_i^*, \forall i; 2 \leq i \leq n$.

**Approximation Algorithm**

In our algorithm, we 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 its previous task) which is not less than $T_i^*$. For instance, the period of the second task, denoted by $T_2$, is determined as

$$T_2 = \left\lceil \frac{T_2}{T_1} \right\rceil T_1 \quad (16)$$

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

$$T_i = \left\lceil \frac{T_i}{T_{i-1}} \right\rceil T_{i-1}. \quad (17)$$

From Eq. (17), it is seen that $T_i \geq T_i^*$ for all $i$. As a result, the total utilization of the system achieved by $T_i$s may be lower 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 we scale the resulting periods so that the total utilization becomes 1. Consequently, the obtained periods satisfy (2c), and hence, comprise a valid solution to the problem.

**Deriving an error bound**

The intuition is that we obtain an upper bound for the weighted sum of the periods returned by Algorithm 1 as a function of $T_i^*$s. Then, since we already have a lower bound for the optimal solution in terms of $T_i^*$ (Eq. (15)), we can compare these two bounds (e.g. by dividing the upper bound by the lower bound) and obtain an upper bound for the error of the presented (approximation) algorithm. For this purpose, we first show that $T_i \leq 2T_i^*, \forall i \in \{1, \ldots, n\}$.
Algorithm 1: Period Assignment

\begin{verbatim}
1 begin
2 \text{\textbf{input}: A WCET } C_i \text{ and a weight } w_i \text{ for each task } \tau_i.
3 \text{// Indexing is done such that } C_i/w_i \leq C_{i+1}/w_{i+1},
4 \text{1} \leq i < n.
5 \text{\textbf{output}: A set of harmonic periods } T_i, 1 \leq i \leq n.
6 \begin{array}{l}
7 \sigma \leftarrow \sum_{1 \leq i \leq n} \sqrt{w_i C_i};
8 \text{for } i \leftarrow 1 \text{ to } n \text{ do}
9 \quad T_i^* \leftarrow \sqrt{C_i/w_i \sigma};
10 \quad T_i \leftarrow T_i^*;
11 \text{\textbf{end}}
12 \text{// Scaling phase}
13 \quad u \leftarrow \sum_{1 \leq i \leq n} C_i/T_i;
14 \text{for } i \leftarrow 1 \text{ to } n \text{ do}
15 \quad T_i \leftarrow u T_i;
16 \text{\textbf{end}}
17 \end{array}
18 \text{// The final result is stored in } T_i, 1 \leq i \leq n.
19 \end{verbatim}

Lemma 3. For any period } T_i \text{ obtained from the approximation algorithm, we have

\[ T_i \leq 2T_i^* \] \hspace{1cm} (18)

Proof. We first show that this relation holds when the periods are assigned using Eq. (17) without the scaling phase. Then, as scaling does not increase the value of the periods, it is concluded that the relation (18) holds for the periods after scaling, too.

The proof is by induction. The base case \((i = 1)\) is trivial since \(T_1 = T_1^* \leq 2T_1^*\). Also, for \(i = 2\), we have

\[ T_2 = \left[ \frac{T_2^*}{T_1^*} \right] T_1 \]

\[ < \left( \frac{T_2^*}{T_1^*} + 1 \right) T_1 \]

\[ = T_2^* + T_1 = T_2^* + T_1 \]

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

\[ T_2 = T_2^* + T_1 \]

\[ \leq T_2^* + T_2^* = 2T_2^* \]

Now, assuming that (18) 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 \(T_i\).

\begin{itemize}
  \item \textbf{case 2: } \(T_{i+1}^* \leq T_i\). In this case, we have \(\left[ \frac{T_{i+1}^*}{T_i^*} \right] = 1\); hence, \(T_{i+1} = T_i\) (due to (17)). According to the assumption \(T_i \leq 2T_i^*\), and since \(T_i^* \leq T_{i+1}\), it follows that

\[ T_{i+1} = T_i \leq 2T_i^* \leq 2T_{i+1}^* \] \hspace{1cm} (19)

\end{itemize}

\[ T_{i+1} = \frac{T_{i+1}^*}{T_i} T_i < \left( \frac{T_{i+1}^*}{T_i} + 1 \right) T_i \]

\[ = T_{i+1}^* + T_i < T_{i+1}^* + T_{i+1} = 2T_{i+1}^* \]

which completes the induction. \(\square\)

Lemma 4. The relative error of Algorithm 1 is not larger than \(2\).

Proof. According to Lemma 3, we have \(T_i \leq 2T_i^*\), and thus, \(w_i T_i \leq 2w_i T_i^*\). Therefore,

\[ \sum_{1 \leq i \leq n} w_i T_i \leq 2 \sum_{1 \leq i \leq n} w_i T_i^* , \]

which results in

\[ \frac{\sum_{i=1}^{n} w_i T_i}{\sum_{i=1}^{n} w_i T_i^*} \leq 2. \]

It means that the relative error of Algorithm 1 cannot be larger than \(2\). \(\square\)

4.3 Error Bound for Equal Weights

In the previous subsection, we showed that the harmonic periods revealed by Eq. (17) satisfy

\[ T_i^* \leq T_i \leq 2T_i^* , \] \hspace{1cm} (20)

This result was obtained for the period values \(T_i\) without applying the scaling phase (i.e., without multiplying by the utilization in Lines 11 to 13 of Algorithm 1). However, such scaling can yield smaller periods (i.e., better results). In this section, we show that the ultimately calculated periods (i.e., after scaling) provide a solution which is not larger than \(2 - 1/n\) times the optimal one, given that equal weights are assigned to the tasks (i.e., \(w_i = w_j\) for all \(i, j\)). Before proceeding with the details, we present a relation which is used in the subsequent discussions.

Proposition 1. Assume that \(x_1, x_2, \ldots, x_n\) denote a set of \(n\) positive integers. Then, the following relation holds

\[ \frac{\sum_{i=3}^{n} x_1}{\left( \sum_{1 \leq i \leq n} \sqrt{x_i} \right)^2} \geq \frac{1}{n} \] \hspace{1cm} (21)

Proof. To verify this, one can define an optimization problem for the left-hand-side of the relation in which the numerator, i.e., \(\sum_{1 \leq i \leq n} x_i\), is regarded as a fixed value and the goal is to maximize the denominator, i.e., \(\left( \sum_{1 \leq i \leq n} \sqrt{x_i} \right)^2\).

Let \(\sum_{1 \leq i \leq n} x_i = X\) where \(X\) is a constant. Therefore, we can write

\[ x_1 = X - \sum_{2 \leq i \leq n} x_i \] \hspace{1cm} (22)

according to which the denominator can be rewritten as \(D^2\) where

\[ D = \sqrt{X - \sum_{2 \leq i \leq n} x_i} + \sum_{2 \leq i \leq n} \sqrt{x_i} \] \hspace{1cm} (23)
Then, the derivative of the denominator with respect to an arbitrary \( x_k \) for \( 2 \leq k \leq n \) is computed as
\[
\frac{\partial (D^2)}{\partial x_k} = 2D \left( \frac{-1}{2\sqrt{X - \sum_{2 \leq l \leq n} x_l}} + \frac{1}{2\sqrt{x_k}} \right).
\]
By putting \( \frac{\partial (D^2)}{\partial x_k} = 0 \) we obtain
\[ x_k = X - \sum_{2 \leq l \leq n} x_l \]
or equivalently, \( x_k = x_1 \). It is also seen that the derivative is positive for \( x_k < x_1 \) and negative for \( x_k > x_1 \). As a result, the function has a maximum at \( x_k = x_1 \). As the computation is valid for any \( x_k \) for \( 2 \leq k \leq n \), we conclude that the function is maximized when \( x_1 = x_2 = \ldots = x_n \).

Besides, as we assumed \( \sum_{1 \leq l \leq n} x_l = X \), it is concluded that \( x_k = X/n \) for \( 1 \leq k \leq n \). As a result, the maximum value of the denominator is equal to \( n\sqrt{X/n}^2 = n^2X/n = nX \).

Based on this, we can write
\[
\left( \sum_{1 \leq l \leq n} x_l \right)^2 \leq X/nX = \frac{1}{n}, \quad \text{(24)}
\]
which proves the claim. \( \square \)

Now, we derive a tighter bound on the approximation error.

**Theorem 3.** The ratio between the weighted sum of the periods provided by Algorithm 1 to that of the optimal one is not larger than \( 2 - 1/n \), given that the weights assigned to the tasks are equal (i.e., \( w_i = w_j \), for all \( i, j \)).

**Proof.** We use \( U^* \) to denote the utilization of the task set using \( T_i^* \) values. According to Corollary 1, we know that \( U^* = 1 \). Further, we define \( U_i^* = \frac{C_i}{T_i^*} \) and \( U_i = \frac{C_i}{T_i} \). The total system utilization after selecting \( T_i \) instead of \( T_i^* \) can be written as
\[
U = \sum_{l=1}^{n} U_l = 1 - U^* + \sum_{l=1}^{n} U_l = 1 + \sum_{l=1}^{n} \left( \frac{C_i}{T_i} - \frac{C_i}{T_i^*} \right).
\]

Now, we scale \( T_i \) through multiplying by \( U \) as
\[
T_i^* = UT_i = T_i + T_i \sum_{l=1}^{n} \left( \frac{C_i}{T_i} - \frac{C_i}{T_i^*} \right) \quad \text{(25)}
\]

For simplifying the above relation, we notice that
\[
\forall l : \frac{C_i}{T_i} - \frac{C_i}{T_i^*} \leq 0 \quad \text{(26)}
\]

As a result, for all \( i \), \( 1 \leq i \leq n \), we can write:
\[
\sum_{l=1}^{n} \left( \frac{C_i}{T_i} - \frac{C_i}{T_i^*} \right) = \sum_{1 \leq l \leq n} \left( \frac{C_i}{T_i} - \frac{C_i}{T_i^*} \right) + \frac{C_i}{T_i} - \frac{C_i}{T_i^*} \leq \frac{C_i}{T_i} - \frac{C_i}{T_i^*}. \quad \text{(27)}
\]

Hence, we can simplify (25) as
\[
T_i^* = T_i + T_i \sum_{l=1}^{n} \left( \frac{C_i}{T_i} - \frac{C_i}{T_i^*} \right) \leq T_i + T_i \left( \frac{C_i}{T_i} - \frac{C_i}{T_i^*} \right) \leq T_i + C_i - \frac{T_iC_i}{T_i^*} \quad \text{(28)}
\]

From (20), \( T_i \) can be written as
\[
T_i = (1 + \alpha)T_i^*, \quad \text{(29)}
\]

where \( 0 \leq \alpha \leq 1 \). As a result, from (28) one can conclude
\[
T_i^* \leq (1 + \alpha)T_i^* + C_i - (1 + \alpha)C_i = (1 + \alpha)T_i^* - \alpha C_i \quad \text{(30)}
\]

The right-hand-side of this relation is maximized (which provides the largest (worst) upper bound for \( T_i^* \)) when \( \alpha = 1 \), which leads to
\[
T_i^* \leq 2T_i^* - C_i \quad \text{(31)}
\]

Consequently, the following relation holds for the total sum of the (scaled) periods
\[
S = \sum_{i=1}^{n} T_i^* \leq \sum_{i=1}^{n} (2T_i^* - C_i) \quad \text{(32)}
\]

Now, we calculate an upper bound for the approximation error through dividing \( S \) by \( S^* \) as
\[
\frac{S}{S^*} \leq \frac{\sum_{i=1}^{n} (2T_i^* - C_i)}{\sum_{i=1}^{n} T_i^*} = 2 - \frac{\sum_{i=1}^{n} C_i}{\sum_{i=1}^{n} T_i^*} \quad \text{(33)}
\]

From (14) we know that (by putting \( w_i = 1 \) for \( 1 \leq i \leq n \))
\[
\sum_{i=1}^{n} T_i^* = \sum_{1 \leq i \leq n} \sqrt{C_i} \sum_{1 \leq i \leq n} \sqrt{C_i^*} = \left( \sum_{1 \leq i \leq n} \sqrt{C_i} \right)^2 \quad \text{(34)}
\]

Using this equation in (33), we will reach
\[
\frac{S}{S^*} \leq 2 - \frac{\sum_{i=1}^{n} C_i}{\left( \sum_{1 \leq i \leq n} \sqrt{C_i} \right)^2}. \quad \text{(35)}
\]

According to Proposition 1, the right-hand-side of this relation is not larger than \( 2 - 1/n \), which yields
\[
\frac{S}{S^*} \leq 2 - \frac{1}{n}. \quad \text{(36)}
\]

\( \square \)

### 4.4 A More Effective Approximation Algorithm

The approximation algorithm introduced in Section 4.2 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 $T_3 \leftarrow T_3^*$ 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. This idea has been originally used by Han et al. in [5] for DCT algorithm. However, their work was to find the largest harmonic period which is smaller than the original given periods of the tasks.

### Algorithm 2: DCT-Based Period Assignment

```plaintext
define input : A WCET $C_i$ and a weight $w_i$ for each task $\tau_i$.

// 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   $\sigma_{\text{min}} \leftarrow \text{null}$;
3   for $i \leftarrow 1$ to $n$ do
4     $T_i^* \leftarrow T_i^*$;
5     for $j \leftarrow i + 1$ to $n$ do
6       $T_j^* \leftarrow \lfloor T_j^*/T_j^* - 1 \rfloor T_j^* - 1$;
7     end
8   for $j \leftarrow i - 1$ down to 1 do
9     $T_j^* \leftarrow T_j^*/T_j^* + 1$;
10    end
11   // Scaling phase
12   $u = \sum_{1 \leq i \leq n} C_i/T_i^*$;
13   for $i \leftarrow 1$ to $n$ do
14     $T_i^* \leftarrow uT_i^*$;
15   end
16   $\sigma \leftarrow \sum_{1 \leq i \leq n} w_i T_i^*$;
17   if $\sigma < \sigma_{\text{min}}$ or $\sigma_{\text{min}} = \text{null}$ then
18     $\sigma_{\text{min}} \leftarrow \sigma$;
19     $T \leftarrow T^*$;
20   end
21 end
```

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 13. Consequently, the upper bound of the error of Algorithm 2 with respect to the optimal period assignment is the same as 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 a polynomial-time algorithm while Algorithm 1 is linear-time.

## 5. EXPERIMENTAL RESULTS

In this section, we evaluate the performance and effectiveness of our proposed approximation algorithms presented in Section 4. In the first subsection we evaluate the algorithms using synthetic task sets while in the next subsection we use benchmark control applications.

### 5.1 Synthetic Task Sets

In this set of experiments we compare our two approximation algorithms with an optimal solution based on an exhaustive search over all possible harmonic period assignments. This optimal algorithm returns a harmonic period assignment with utilization $1$ which minimizes (2a). The optimal algorithm (derived from the Forward approach in [10]) 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 added a pruning rule to limit the results to assignments with $U \leq 1$. In the experiments we measure the relative error of the proposed algorithms as

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

where $T_i^*$ is obtained from (14). 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 order to generate random task sets, we must generate random values of WCET. In the first experiment we consider the effect of relative WCET ratio, i.e., $C_i/C_{i-1}$. In this experiment we consider 7 tasks. The WCET of each task has been randomly selected with uniform distribution so that $C_i \in [C_{i-1}, kC_{i-1}]$ where $C_i \in [1, 10]$. For each parameter value, 1000 random task sets have been generated. As it has been 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 $T_i^*$. Since valid assignments must be larger than $T_i^*$ values, if $T_i^* / T_i^* \approx 1$, Algorithm 1 assigns $T_2 \leftarrow 2T_i$ which will be much larger than $T_i^*$. However, since $T_i^*$ 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 $T = \{T_1^*, 2T_1^*, 2T_i^*, \ldots, 2T_i^*\}$ while the period assignment from Algorithm 2 is $T^* = \{T_1^*, T_2^*, \ldots, T_n^*\}$. It happens because if we have small ratio between consecutive $C_i$ values, then $T_i^* < 2T_i^*$. In the next experiment, we study the effect of selection
In the last experiment we consider the effect of the number of tasks, i.e., $n$. The WCET of each task has been selected randomly with uniform distribution from the interval of $[1,500]$ while $n$ varies from 5 to 30 tasks. As it can be seen in Fig. 4, even with the increase in $n$, the relative error of the algorithms will not increase even though according to Section 4.3, we expect that the error bound of Algorithm 1 would be $E \leq 2 - 1/n$. Moreover, when $n$ increases, the number of values selected from the range increase which in turn increases $\sum \sqrt{w_i C_i}$. Thus the same effect that happened for the large $\sigma$ values in Fig. 3 happens here as well.

5.2 Experiments on Control Systems

In order to evaluate our methods with respect to control applications, we define the following three plant (i.e., physical processes under control) families:

- **Family I**: All plants have two stable poles and are drawn from $P_1(s)$ and $P_2(s)$ with equal probability where

  $$P_1(s) = \frac{1}{(s+a_1)(s+a_2)}$$
  $$P_2(s) = \frac{1}{s^2 + 2\zeta \omega s + \omega^2}$$

  with $a_1, a_2, \zeta \in \text{unif}(0,1), \omega \in \text{unif}(0,1)$.

- **Family II**: All plants have two stable or unstable poles, with each plant drawn with equal probability from

  $$P_3(s) = \frac{1}{(s+a_1)(s+a_2)}$$
  $$P_4(s) = \frac{1}{s^2 + 2\zeta \omega s + \omega^2}$$

  with $a_1, a_2, \zeta \in \text{unif}(-1,1), \omega \in \text{unif}(0,1)$.

- **Family III**: All plants have three stable or unstable poles, with each plant drawn with equal probability from

  $$P_5(s) = \frac{1}{(s+a_1)(s+a_2)(s+a_3)}$$
  $$P_6(s) = \frac{1}{(s^2 + 2\zeta \omega s + \omega^2)(s+a_3)}$$

  with $a_1, a_2, a_3, \zeta \in \text{unif}(-1,1), \omega \in \text{unif}(0,1)$.

We randomly generated 10 sets of plants for each family and designed LQG (Linear Quadratic Gaussian) controllers to minimize the cost

$$J_i = \lim_{t \to \infty} \frac{1}{t} \mathbb{E} \int_0^t \left( x_i^T Q_{1i} x_i + u_i^T Q_{2i} u_i \right) \, dt$$

(37)

for each plant, where $x_i$ is the plant state and $u_i$ is the control signal. LQG control is a fundamental optimal control strategy. An LQG controller is used to minimize the running cost of the state and input signal.

We used the design parameters $Q_{1i} = BB^T$, and $Q_{2i} = 0.01I$ ($Q_{1i}$). The task execution times were randomly generated from $C_i \in \text{unif}(0.01,0.1)$. The weights were randomly chosen from $\omega_i \in \text{unif}(0,1)$. Task 1 has the highest priority, while task 3 has the lowest priority.

We first calculated the response time distribution for each task. The distributions were truncated from below if the response time variation was larger than the period. Using
the response time distributions, stochastic LQG controllers were designed. Then the overall LQG cost was evaluated. The procedure above provides a way to calculate the LQG cost by knowing the periods. Finally the Sequential Search method was used to obtain the local optimal LQG cost and the corresponding (non-harmonic) periods. The details of the initial period assignment can be found in [27].

Based on the initial period values, we use Algorithm 1 and 2 to find harmonic periods and then evaluate the LQG control performance. The LQG controllers were designed using the Jitterbug toolbox [26], and the evaluation was done using the TrueTime toolbox [26]. For each family, the average costs are given in Table 1 (smaller values are better).

Table 1: Control costs for different methods.

<table>
<thead>
<tr>
<th>Family</th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-harmonic</td>
<td>6.80</td>
<td>19.67</td>
<td>17.04</td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>3.07</td>
<td>6.78</td>
<td>15.59</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>2.91</td>
<td>6.71</td>
<td>15.17</td>
</tr>
</tbody>
</table>

In Table 1, we design the LQG controllers and evaluate the LQG costs as follows:

- **Non-harmonic.** The delay distribution is truncated to the interval \([R^\text{best}_i, R^\text{best}_i + T_i]\). \(R^\text{best}_i\) is the best case response time. We calculated first 1000 job response times, and choose the smallest one as the best case response time. The probability of a response time greater than \(R^\text{best}_i + T_i\) is added to the probability mass function at \(R^\text{best}_i + T_i\). We then use the truncated delay distribution to design a stochastic LQG controller. The LQG cost is evaluated using TrueTime.

- **Algorithm 1.** We assign the harmonic periods using Algorithm 1. LQG controllers with constant delays equal to \(R_i\) are designed in Jitterbug and evaluated in TrueTime.

- **Algorithm 2.** We assign the harmonic periods using Algorithm 2. LQG controllers with constant delays equal to \(R_i\) are designed and evaluated.

As seen in Table 1, for all three families of plants, the costs incurred by harmonic periods are smaller than that of the initially selected non-harmonic ones. The reason is that in harmonic task sets, if the execution time does not vary at run-time, then there will be no jitter in sampling and actuation. It is also observed that Algorithm 2 is better than Algorithm 1, because Algorithm 2 provides harmonic periods with smaller approximation error, although the computational complexity of Algorithm 2 is larger.

### 6. DISCUSSIONS

Table 2 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.

<table>
<thead>
<tr>
<th>max (U)</th>
<th>min (\sum w_iT_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>restricted range</td>
<td>NP-hard in the weak sense (at least)</td>
</tr>
<tr>
<td>unrestricted range</td>
<td>There exists poly. time algorithm</td>
</tr>
</tbody>
</table>

\(\sum w_iT_i\). If periods are not restricted, i.e., they can be any value, then one possible harmonic assignment is

\[T_1 = T_2 = \ldots = T_n = \sum_{i=1}^n C_i\] (38)

In (38), period ratio between any two consecutive periods is 1 while the utilization is 1. Hence it is a linear-time solution for Problem 1 when periods are not restricted. Through Section 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.

Regarding the tightness of the obtained error bounds for Algorithm 1 (which are also valid for Algorithm 2), we did not find an example with exact error 2 (or 2 – 1/n for equal weights). Thus, it remains as an open question whether a better (tighter) error bound exists for the presented approximation algorithms.

Another interesting observation in our experiments is that 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 or not. If it is the case, then Problem 2 can be solved in polynomial-time using Algorithm 2.

### 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. We have also considered the problem of minimizing the weighted sum of harmonic periods where there is no upper bound on the valid periods. We presented two polynomial-time approximation algorithms which are able to assign harmonic periods while minimizing the total sum of periods. 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 2. 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 2, the first algorithm has an error which is smaller than 0.06. The second algorithm is as good as the optimal solution based on an exhaustive search.

As our future work, we investigate the optimality of our
second algorithm. 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 will provide a new method to obtain the space of feasible periods for RM.

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

**8. REFERENCES**


