A Framework to Construct Customized Harmonic Periods for Real-Time Systems

Mitra Nasri
School of Electrical and Computer Engineering, University of Tehran, Iran
mitra.nasri@ut.ac.ir

Gerhard Fohler
Chair of Real-time Systems, Technische Universität Kaiserslautern, Germany
fohler@eit.uni-kl.de

Mehdi Kargahi
School of Electrical and Computer Engineering, University of Tehran, Iran
kargahi@ut.ac.ir

***Abstract*—The periodic task model has been widely used in real-time systems due to the periodic behavior of many applications or periodic observation patterns of environmental events as in control applications. While the tasks of some applications have inherent values for periods, many can be defined via *ranges* of acceptable values. The designer choice of period values has consequences w.r.t. to utilization of the task set and the resulting hyperperiod. Harmonic task sets are favored, e.g., for their polynomial-time worst case response time analysis or their small hyperperiods, which is of major concern e.g., for hypervisors used in virtualization or time triggered systems.**

In this paper we present a model to describe harmonic relations between ranges of period values, rather than single numbers only. We derive sufficient conditions for the existence of a linear-time solution, as well as a graph representation for the relations between period ranges. We provide utilization bounds of each resulting harmonic range, giving the designer flexibility to select a harmonic task set with high or low utilization. The tightness of the bounds as well as efficiency of our period assignment algorithms have been evaluated by synthetic experiments via system utilization and feasibly constructed harmonic task sets.

I. Introduction

The periodic task model has been widely used in real-time systems due to periodic behavior of many applications or periodic observation patterns of environmental events as in control applications. Often, deadlines are considered implicit, i.e., as long as the periodic behavior is guaranteed, certain levels of safety and/or quality of service (QoS) are satisfied [1]. In such systems, there is usually a trade-off between period and QoS level which is addressed by providing a range of periods for the tasks representing the applications.

A large number of real-time scheduling algorithms have been presented for periodic tasks, such as the earliest deadline first (EDF) and rate monotonic (RM) [2]. The exact schedulability analysis of RM is an NP-complete problem [3] unless the periods are harmonic (simply periodic) [4], i.e., each period is an integer multiplier of the smaller periods. According to [5] and [6], in systems with a large number of tasks or tasks with period ranges, finding harmonic relations between periods has pseudo-polynomial computational complexity and highly depends on the largest period value. When the utilization of the system is taken into account, this problem becomes more complicated since period assignment affects system utilization and may make the system even infeasible.

Harmonic periods have been demanded in a large spectrum of industrial real-time applications such as avionics, submarines, and robotics ([7], [8], [9], [10]) as well as control systems with nested feedback loops [11]. RM can schedule harmonic tasks up to 100% utilization [4], the worst-case response-time (WCRT) of the tasks can be found in polynomial-time [12]. Task sets with harmonic periods have small hyperperiods [13], [5], which is a major concern for hypervisors in virtualization [14] or time triggered systems [15].

In this paper we consider periodic task sets with *ranges* for the periods of individual tasks. The basic contribution of the work is the derivation of a model to describe harmonic relations between ranges of period values, not between discrete numbers. It means that we find sub-intervals within given period ranges of the task set which are in harmonic relations with each other. To find those sub-intervals, the task set is represented by a graph which is based on the existence of harmonic relations between period ranges. Using this graph, it is also possible to construct customized harmonic relations, e.g., periods which are 2 times of the smaller period, by restricting non-favorable multipliers.

We derive sufficient conditions for the existence of a linear-time solution as well as the utilization bound of each resulting harmonic range, giving the designer flexibility to select a harmonic task set with high or low utilization, e.g., having high system utilization for maximizing quality of control [1] or QoS [16]. These bounds also help designers to try different period ranges when a desired solution cannot be found due utilization constraints.

In this paper we provide a theoretical framework allowing designers to change input period ranges such that a feasible harmonic relation and respective utilization bound is achievable. At the same time we provide means to construct *customized period relations* (which is useful for applications such as dwell tasks in Radars [9]). Using that graph, any given set of preferences for the relation between periods can be provided by pruning the edges. On the other hand, while finding harmonic periods for the task set is generally a problem with pseudo-polynomial complexity, we provide sufficient conditions for the existence of a linear-time solution.

Previously, Han and Tyan [4] have presented an algorithm called *Sr* to map the original periods to the largest artificial harmonic periods which are smaller than the original ones by

an algorithm called Sr. Their goal was to analyze schedulability of the system; if all tasks could map to the artificial periods while the utilization is not more than 1, the RM schedulability is guaranteed. In Sr algorithm, artificial periods are powers of 2. In [17], harmonic deadlines have been created using least common multipliers (LCM) of the previous tasks, to verify schedulability of the systems with deadlines larger than periods. In these studies, period ranges are not considered, and also tasks are executed by their original non-harmonic periods. On the other hand, [18] and [9] used artificial harmonic periods created by [4] methods instead of original periods of the tasks in Radar systems. In [18], they also tried to find minimum number of harmonic chains, i.e., subsets of tasks with harmonic relations, thus the focus of [18] was not totally on fully harmonic task sets. Also in [19], Sr algorithm has been applied to find partitions of tasks with harmonic periods for multiprocessor scheduling. However, none of these contributions considers ranges of periods.

There have been other studies to assign periods to the tasks from given period ranges to satisfy other objectives such as maximizing quality of control as in [1] and [20]. However these works are not based on extracting harmonic periods, nor do they provide utilization bound of the given solution for the designers. For example, by period transformation method introduced in [21] it is possible to reserve the processor for a task which may have nondeterministic release times in a periodic fashion. Although this model has capability to transform periods, it has never been used to construct harmonic periods. Also tasks are not assumed to have period ranges.

The closest contribution to the one presented here is [5] which aims at minimizing hyperperiods. They use integer multipliers of each period range to find an intersection between all period ranges, which is the hyperperiod. They noticed that intervals generated by multiplying an integer number by a period range, become wider and wider until they overlap. We also considered this situation. Although it is not possible to use their method to obtain harmonic periods, they were among the firsts to consider period ranges. They did not consider utilization of the tasks during the period assignment phase as they assumed period ranges are short and the utilization is not changed by considering different periods for the tasks. They present one condition to bound the hyperperiod value, while we present 2 more to check the existence of tighter bounds for that, e.g., the cases were period ranges or multipliers of them, totally cover period range of another task. Finally, their performance to find a solution is pseudo-polynomial and is highly dependent on the given period ranges while we present conditions for which the problem is solvable in linear-time.

The reminder of the paper is organized as follows; Sect. II introduces the system model. Sect. III presents a model to derive harmonic relations. A graph representation of the harmonic relations in the task set is presented in Sect. IV and we show how it is possible to derive harmonic relations in the task set. In Sect. V, utilization bounds of a solution has been derived and two greedy period assignment algorithms are introduced. Sect. VI presents experimental results and finally the paper is concluded in VII.

II. System Model

We consider a single processor system and hard real-time task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ with n independent tasks. Each task is identified by $\tau_i : (c_i, T_i^{min}, T_i^{max})$, where c_i is the worst-case execution time (WCET), and $T_i^{min}, T_i^{max} \in \mathbb{R}^+$ are the minimum and the maximum values for period of task τ_i which are given by the designer. It is assumed that tasks are indexed according to the starting point of the period ranges, i.e., $T_i^{min} \leq T_{i+1}^{min}$, $1 \leq i < n$. Also tasks have no release-offset and they assumed to have periodic behavior with implicit deadline which is equal to their period. As mentioned before, many of the control applications as well as multimedia applications have such property.

The goal of the paper is to assign one period value T_i for each task τ_i, $1 \leq i \leq n$ such that the task set become harmonic, i.e., $T_j = k_j \times T_{j-1}$ for $1 < j \leq n$ where $k_j \in \mathbb{N}^+$ is an integer number greater than or equal to 1.

III. A Model to Describe Harmonic Relations between Period Ranges

In this section, we verify the existence of harmonic relations between two or more period ranges which will be referred as *interval* in this section, and discuss the complexity of finding such relations in general. The results will be used in the next sections to construct fully harmonic task sets.

As the first step, we define a harmonic relation between two intervals I_1 and I_2 which are known by their start and end values as $[I_1^s, I_1^e]$ and $[I_2^s, I_2^e]$. A harmonic relation between I_1 and I_2 exists if it is possible to find $i_1 \in I_1$ and $i_2 \in I_2$ such that $\frac{i_2}{i_1} \in \mathbb{N}$.

We define a projected harmonic zone as:

Definition 1. The projected harmonic zone $\chi^a_{I_1 \to I_2} : [\chi^s, \chi^e]$ from interval I_1 to I_2, $I_1^s \leq I_2^s$, with multiplier $a \in \mathbb{N}^+$, is a range of numbers in I_2 that starts from $\chi^s = max\{I_2^s, aI_1^s\}$ and ends to $\chi^e = min\{I_2^e, aI_1^e\}$, and for any $i_2 \in I_2$ there exists at least one $i_1 \in I_1$ such that $\frac{i_2}{i_1} \in \mathbb{N}$.

The following theorem presents the necessary conditions for existence of harmonic relation between two intervals.

Theorem 1. *The necessary condition for the existence of any harmonic relation between two intervals I_1 and I_2, $I_1^s \leq I_2^s$, is that either $\lfloor \frac{I_2^s}{I_1^e} \rfloor \leq \frac{I_2^e}{I_1^s} - 1$, or $\frac{I_2^e}{I_1^s} = \frac{I_2^s}{I_1^e}$ is true. Otherwise there will be no harmonic relation between any two possible numbers in I_1 and I_2, and hence, for any arbitrary $a \in \mathbb{N}^+$, $\chi^a_{I_1 \to I_2} = \emptyset$.*

Proof. To prove the theorem we show that if none of the two conditions hold, there is not any integer multiplier such as $a \in \mathbb{N}^+$ that constructs a projected harmonic zone from I_1 to I_2. For this aim, we calculate $a^s = max\{\lfloor \frac{I_2^s}{I_1^e} \rfloor, 1\}$ which is the largest possible integer multiplier for I_1 that if it is multiplied by I_1^e (the largest value in I_1), the result is still smaller than I_2^s. Fig. 1 shows a^s with respect to the intervals. As it can be seen in the figure, for any integer value $a < a^s$, there is no harmonic zone from I_1 to I_2 since $aI_1^e < a^s I_1^e \leq \frac{I_2^s}{I_1^e} I_1^e \leq I_2^s$. On the

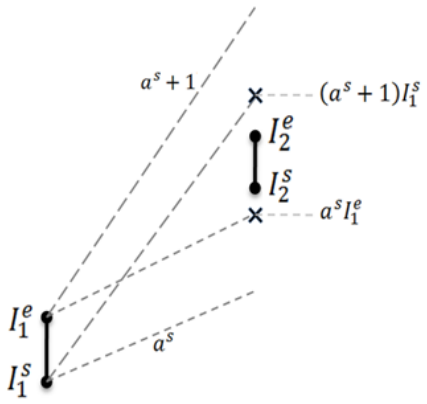

Fig. 1. A case where there can be no harmonic relation between two intervals

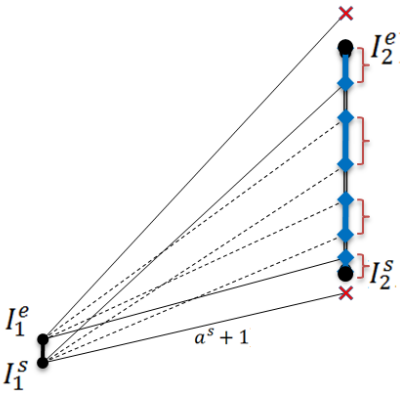

Fig. 2. A case where there are more than one integer multipliers between two intervals

other hand since $a^s = max\{\lfloor \frac{I_2^s}{I_1^e} \rfloor, 1\}$, we have $(a^s+1)I_1^e \geq I_2^s$ and hence, multipliers greater than a^s may have chances to produce projected harmonic zones on I_2. However, if for the next integer multiplier, i.e., a^s+1, the minimum possible value which can be created from I_1, is bigger than I_2^e, there is no harmonic relation that can be projected from I_1 to I_2. In other words, if we have $(a^s+1)I_1^s > I_2^e$ which means that $\lfloor \frac{I_2^s}{I_1^e} \rfloor > \frac{I_2^e}{I_1^s} - 1$, there is no chance of creating harmonic zones from a^s+1, a^s+2 and so on. Also since according to our assumption, $\frac{I_2^e}{I_1^s} \neq \frac{I_2^s}{I_1^e}$, there is no harmonic relation between the smallest value of I_1 and the biggest value of I_2 with multiplier a^s+1. Thus the proof has been completed and as long as none of the necessary conditions holds, projected harmonic zone $\chi^a_{I_1 \to I_2}$ is empty for any $a \in \mathbb{N}^+$. ■

When two intervals I_1 and I_2, $I_1^s \leq I_2^s$, have intersections, i.e., $I_2^s \leq I_1^e$, we have $\lfloor \frac{I_2^s}{I_1^e} \rfloor = 0$. As a result, $\lfloor \frac{I_2^s}{I_1^e} \rfloor \leq \frac{I_2^e}{I_1^s} - 1$ because $0 \leq \frac{I_2^e}{I_1^s} - 1$.

As shown by Fig. 1, if conditions of Theorem 1 hold, there will be a set of multipliers which can project interval I_1 to some subintervals of interval I_2. The following definition describes a set of harmonic multipliers between two intervals.

Definition 2. For two intervals I_1 and I_2, $I_1^s \leq I_2^s$, the set of harmonic multipliers is defined as $A_{I_1 \to I_2} = \{a_1, a_2, \ldots, a_z\}$ where $a_1 = \lfloor \frac{I_2^s}{I_1^e} \rfloor + 1$, $a_2 = a_1 + 1$, $a_3 = a_2 + 1$, …, $a_z = \lfloor \frac{I_2^e}{I_1^s} \rfloor$. If $\lfloor \frac{I_2^s}{I_1^e} \rfloor \in \mathbb{N}$, we define $a_1 = \lfloor \frac{I_2^s}{I_1^e} \rfloor$.

Lemma 1. Definition 2 defines all possible integer multipliers between two intervals I_1 and I_2, $I_1^s \leq I_2^s$.

Proof. We show it is not possible to find integer multiplier $b \in \mathbb{N}$, where $b < a_1$ or $b > a_z$ such that $\chi^b_{I_1 \to I_2} \neq \emptyset$. In other words, it is not possible to use b for projecting $i_1 \in I_1$ to $i_2 \in I_2$ with harmonic relation $i_2 = b \times i_1$. For this aim, we consider the minimum and maximum values of I_1 to create a possible range for i_2 as $[bI_1^s, bI_1^e]$, then in the following steps we show that this bound has no intersection with I_2, and hence, such i_2 does not exist. In the first step we assume $b > a_z$, thus $b > \lfloor \frac{I_2^e}{I_1^s} \rfloor$ which means that $b \geq \lfloor \frac{I_2^e}{I_1^s} \rfloor + 1$. Now according to the bounds of i_2 we have:

$$i_2 \in \left[\left(\left\lfloor \frac{I_2^e}{I_1^s} \right\rfloor + 1 \right) I_1^s, \left(\left\lfloor \frac{I_2^e}{I_1^s} \right\rfloor + 1 \right) I_1^e \right] \tag{1}$$

However, because of the fact that $\lfloor I_2^e / I_1^s \rfloor + 1 > I_2^e / I_1^s$, we can deduce that $(\lfloor I_2^e / I_1^s \rfloor + 1) I_1^s > I_2^e$, which means that the intersection of (1) and I_2 is empty. In the second case we assume $b < a_1$. We consider two cases; $I_1^e = I_2^s$ and $I_1^e < I_2^s$. In the first case, because of Definition 2, b has to be 0, which is impossible because b has to be greater than 0 to produce positive projections. The second case can be directly proven by the definition of a_s in Theorem 1 because $\lfloor \frac{I_2^s}{I_1^e} \rfloor$ is the largest integer multiplier that if it is applied on I_1^e, the result still will be smaller than I_2^s since $\lfloor \frac{I_2^s}{I_1^e} \rfloor I_1^e < \frac{I_2^s}{I_1^e} I_1^e$. ■

From Lemma 1, we can conclude that if $A_{I_1 \to I_2} = \emptyset$, there is no harmonic multiplier between the two intervals. We denote the set of all harmonic relations between the two intervals by $\chi_{I_1 \to I_2}$ and it is a set of all projected harmonic zones $\chi^{a_1}_{I_1 \to I_2}, \chi^{a_2}_{I_1 \to I_2}, \ldots, \chi^{a_z}_{I_1 \to I_2}$ where a_1 to a_z are defined by Definition 1.

One of the results of Lemma 1 is that if instead of integer multipliers, we search for integer divisors from I_2 to I_1 and try to find subintervals of I_1 which are covered by integer divisors of I_2, still there will be no harmonic relation which cannot be generated by the multiplier set A defined in Definition 2. In other words, $\chi_{I_1 \to I_2}$ covers all possible harmonic relations of two intervals I_1 and I_2 regardless of the order of the intervals. This important outcome is concluded in the following corollary.

Corollary 1. *For two intervals I_1 and I_2, $I_1^s \leq I_2^s$, all possible pairs of $i_1 \in I_1$ and $i_2 \in I_2$ which have harmonic relation with each other, can be covered by at least one harmonic multiplier in A according to Definition 2.*

Proof. To prove this claim we consider two cases; $I_1^e < I_2^s$ and

$I_1^e \geq I_2^s$. In the first case, the only possible set of harmonic relations can be obtained by finding integer divisors which map some values of I_2 to some values of I_1. Each integer divisor from I_2 to I_1 is in fact an integer multiplier from I_1 to I_2, which is considered in set A of multipliers because of Lemma 1. The only chance that there might be new integer multipliers from I_2 to I_1 is when these two intervals have intersections. This situation has been considered in the second case where $I_1^e \geq I_2^s$. However, since in that case, $a_1 = \lfloor \frac{I_2^s}{I_1^e} \rfloor + 1 = 1$, we can see that the intersection $[I_2^s, I_1^e]$ is also covered by A with multiplier 1. Hence, the order of the intervals does not affect the existence of harmonic relations between the intervals which are defined by $\chi_{I_1 \to I_2}$. ■

In the next step we extend the definition of harmonic relations to set of m intervals.

Definition 3. For the set of intervals I_1, I_2, ..., I_m, harmonic relation $\chi_{I_1 \to I_2 \to \ldots \to I_m}$ exists if it is possible to find $i_1 \in I_1$, $i_2 \in I_2$, ..., $i_m \in I_m$ such that $\frac{i_2}{i_1} \in \mathbb{N}$, $\frac{i_3}{i_2} \in \mathbb{N}$, ..., $\frac{i_m}{i_{m-1}} \in \mathbb{N}$.

Since in Definition 3, the value $i_j \in I_j$, $1 < j \leq m$ can be constructed from multiplying an integer number by $i_{j-1} \in I_{j-1}$, according to Lemma 1 it has to belong to $\chi_{I_{j-1} \to I_j}$. It gives us a hint about how we can verify the existence of harmonic relations between a set of intervals. Consider an example with 3 intervals $I_1 : [11, 14]$, $I_2 : [20, 49]$, and $I_3 : [30, 40]$. To verify the existence of $\chi_{I_1 \to I_2 \to I_3}$, we have to find $\chi_{I_1 \to I_2}$. In this example we have $\chi^2_{I_1 \to I_2} = [22, 28]$, $\chi^3_{I_1 \to I_2} = [33, 42]$, and $\chi^4_{I_1 \to I_2} = [44, 49]$. Then we can check the existence of projected harmonic zones between any of those zones and the target interval I_3 which leads to $\chi^1_{\chi^3_{I_1 \to I_2} \to I_3} = [33, 40]$.

As shown by Definition 2, there might be more than one multiplier between two intervals. The number of multipliers can be large depending on the input. For example for two intervals $I_1 : [70, 72]$, $I_2 : [140, 5039]$, there are 68 harmonic projected zones, 32 of them have no intersections ($\chi^2_{I_1 \to I_2}$, $\chi^3_{I_1 \to I_2}$, ..., $\chi^{34}_{I_1 \to I_2}$) and 36 of them create a big continuous coverage of projected zones on the I_2 ($\chi^{35}_{I_1 \to I_2}$, $\chi^{36}_{I_1 \to I_2}$, ..., $\chi^{71}_{I_1 \to I_2}$). Fig. 3 shows one symbolic example for such a verification process.

If $\chi_{I_1 \to I_2}$ creates only one projected harmonic zone, existence of $\chi_{I_1 \to I_2 \to I_3}$ can be verified in one step by calculating $A_{\chi_{I_1 \to I_2} \to I_3}$. The following lemma shows an important property about two continuous projected harmonic zones.

Lemma 2. If for two intervals I_1 and I_2, $I_1^s \leq I_2^s$, two adjacent projected harmonic zones $\chi^a_{I_1 \to I_2}$ and $\chi^{a+1}_{I_1 \to I_2}$ for $a \in A_{I_1 \to I_2}$, $a_1 \leq a \leq a_z$, are continuous (have non-empty intersections), all other projected harmonic zones $\chi^b_{I_1 \to I_2}$ and $\chi^{b+1}_{I_1 \to I_2}$ where $b \in \{a, a+1, \ldots, a_z - 1\}$ produce continuous intervals.

Proof. If $\chi^a_{I_1 \to I_2}$ and $\chi^{a+1}_{I_1 \to I_2}$ are continuous, aI_1^e which is the ending point of $\chi^a_{I_1 \to I_2}$, should be greater than or equal to $(a+1)I_1^s$, which is starting point of $\chi^{a+1}_{I_1 \to I_2}$. Thus we have:

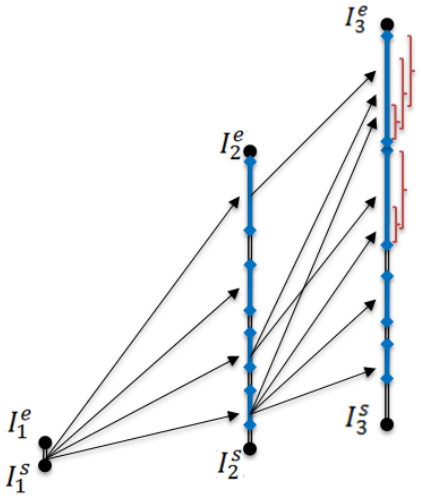

Fig. 3. A case where the number of projected zones can be dramatically large

$$(a+1)I_1^s \leq aI_1^e \Rightarrow (a+1)I_1^s + I_1^s \leq aI_1^e + I_1^e \Rightarrow$$
$$(a+2)I_1^s \leq (a+1)I_1^e \quad (2)$$

As a result, the starting point of $\chi^{a+2}_{I_1 \to I_2}$ is also greater than or equal to the ending point of $\chi^{a+1}_{I_1 \to I_2}$, which means that they are continues. This situation occurs for all other multipliers greater than a. ■

It is important to mention that the existence of one continuous harmonic projected zone between pairs of intervals, does not necessarily confirm the existence of harmonic relations between all of them. For example assume $I_1 : [50, 52]$, $I_2 : [51, 60]$, and $I_3 : [58, 63]$. Then $X_{I_1 \to I_2} : [51, 52]$, $X_{I_2 \to I_3} : [58, 60]$, but $X_{I_1 \to I_3} = \emptyset$ since $A_{I_1 \to I_3} = \emptyset$. It means that there will be no integer multiplier which is able to map some values of I_1 to I_3. As a result, $\chi_{I_1 \to I_2 \to I_3} = \emptyset$.

Definition 4. Two intervals I_1 and I_2, $I_1^s \leq I_2^s$ have a tight harmonic relation, denoted by $X_{I_1 \to I_2}$, if $\chi_{I_1 \to I_2}$ produces a continuous projected harmonic zone which covers any value from I_2^s to I_2^e.

The following theorem defines the necessary and sufficient conditions of existence of tight harmonic relation between two intervals.

Theorem 2. *Validity of at least one of the following conditions is a necessary and sufficient condition for the existence of a tight harmonic relation between two intervals I_1 and I_2, $I_1^s \leq I_2^s$.*

$$a_1(I_1^e - I_1^s) \geq I_1^s \quad (3)$$
$$\frac{I_1^s I_1^e}{I_1^e - I_1^s} \leq I_2^s \quad (4)$$
$$I_2^e \leq I_1^e \quad (5)$$
$$a_1 I_1^s \leq I_2^s \leq I_2^e \leq a_1 I_1^e \quad (6)$$

Proof. First we discuss sufficiency of these conditions. According to (3), $a_1 I_1^e$ which is the ending point of $\chi_{I_1 \to I_2}^{a_1}$, is greater than or equal to the starting point of the next harmonic zone at $(a_1+1)I_1^s$. Hence, because of Lemma 2 we know that all other harmonic projected zones are also continuous. On the other hand, condition (4) shows that the starting point of the second interval is large so that in all cases the projections produced from the first interval collide each other. For the proof we start by (3) thus we have $a_1 \geq \frac{I_1^s}{I_1^e - I_1^s}$. Also based on Definition 2, $a_1 = \lfloor \frac{I_2^s}{I_1^e} \rfloor + 1$ as a result we have:

$$\left\lfloor \frac{I_2^s}{I_1^e} \right\rfloor + 1 \geq \frac{I_2^s}{I_1^e} \geq \frac{I_1^s}{I_1^e - I_1^s} \tag{7}$$

which can be easily simplified into (4). On the other hand, when condition (5) holds, $I_2^e \leq I_1^e$ and since we already have $I_1^s \leq I_2^s$, interval I_1 totally covers interval I_2 and a_1 becomes 1. Finally, if A has only one multiplier, i.e., $a_1 = a_z$, there can be only one harmonic projected zone between I_2 and I_1. Consequently, the projected zone will be continuous. If $a_1 I_1^s \leq I_2^s \leq I_2^e \leq a_1 I_1^e$, the resulting projected zone covers I_2.

For necessity of the conditions it is enough to show that if none of them hold, there will be no tight harmonic relation between the intervals. By such assumption we know those two intervals have no intersection (because of not having (5)), and the number of multipliers is more than 1 (because of not having (6)), and I_2^s is not greater than the value from which all projected harmonic zones of I_1 will be continuous (because of not having (3) or (4)). Thus, if $A_{I_1 \to I_2} \neq \emptyset$, there will be at least one gap between the first and the second projected harmonic zones at $[a_1 I_1^e, a_2 I_1^s] \subseteq I_2$ which is not covered by $\chi_{I_1 \to I_2}$. ■

Conditions of Theorem 2 confirm existence of transitive closure property in harmonic projected zones. By this property, we can deduce existence of $X_{I_1 \to I_2 \to I_3}$ knowing $X_{I_1 \to I_2} \neq \emptyset$ and $X_{I_2 \to I_3} \neq \emptyset$. The reason is that existence of $X_{I_1 \to I_2}$ means that any value in I_2 can be obtained by an integer multiplier of I_1, also because of the existence of $X_{I_2 \to I_3}$, we know any value in I_3 can be produced by applying an integer multiplier from I_2. Now we can deduce that any value in I_3 is obtained by applying integer multipliers of I_1.

The coverage property of tight harmonic relations confirms the fact that if we have $X_{I_j \to I_{j+1}} \neq \emptyset$ for $1 \leq j < n$ intervals, all value in those intervals can be in harmonic relations with each other, and hence, we can deduce existence of $X_{I_1 \to I_2 \to \ldots \to I_n}$.

Here after, we use set operator $X_{\{I_1, I_2\}}$ equivalently instead of $X_{I_1 \to I_2}$. Further to show that a harmonic or tight harmonic relation exists between set $S_i \subseteq \tau$ of the tasks, we simply use χ_{S_i} and X_{S_i}, respectively.

IV. Extracting Harmonic Relations

In this section we show how it is possible to find harmonic relations in the task set. For this aim for each task τ_i, $1 \leq i \leq n$ we find sub-interval $I_i \subseteq [T_i^{min}, T_i^{max}]$ such that harmonic relation $\chi_{I_1 \to I_2 \to \ldots \to I_n}$ exists. In other words, the solution will not be one period value for each task but it will be a sub-interval from the original period range where for any arbitrary value in the interval I_i, it is possible to find values with harmonic relations from I_1 to I_n. In the next section we present a simple method to assign periods to the task such that the resulting task set has high or low utilization.

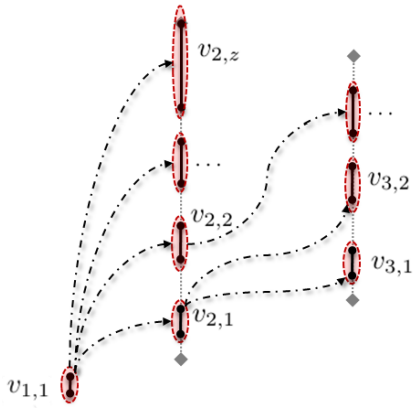

Fig. 4. An example for representing the task set as a graph with harmonic relations

Based on the existence of harmonic zones between period ranges of two consecutive tasks, we define a graph $G : (V, E)$. In this graph, each vertex represents a continuous harmonic zone, and edges represent respective multipliers which are used to create those harmonic zones inside the period range of the next task. An example of the graph is shown in Fig. 4. In this graph, $v_{1,1}$ represents period range of task τ_1. Based on the number of multipliers between τ_1 and τ_2, i.e., $A_{\tau_1 \to \tau_2}$, we will have some projected harmonic zones from τ_1 to τ_2. For example if there are 2 number of multipliers, there will be 2 vertices in the second layer of the graph, denoted by $v_{2,1}$ and $v_{2,2}$ in the figure. Each of these vertices are assigned to a projected harmonic zone denoted by $I_{v_{2,j}}$, $1 \leq j \leq \|A_{\tau_i \to \tau_j}\|$. In the next layer, each interval $I_{v_{2,j}}$ will be a source interval to find possible projected harmonic zones in period range of task τ_3. However, since starting point of $I_{v_{2,j}}$ may be greater than T_3^{min}, we consider interval $\left[max\{T_3^{min}, I_{v_{2,j}}^s\}, T_3^{max}\right]$ instead of $[T_3^{min}, T_3^{max}]$.

As shown by Lemma 2, if for one multiplier such as a_x in set $A_{I_1 \to I_2}$ of multipliers between two intervals I_1 and I_2, two harmonic projected zones are overlapping, the result will be a continuous projected harmonic zone for $\chi_{I_1 \to I_2}^{a_x}$ to $\chi_{I_1 \to I_2}^{a_z}$ which is the last projected zone. Formula (3) verified occurrence of continuous zones from the first first multiplier but we can easily apply it on other multipliers to see from which multiplier, the projected harmonic zones will be continuous.

$$a_x \geq \frac{I_1^s}{(I_1^e - I_1^s)} \tag{8}$$

Thus we are able to replace all projected harmonic zones produced by a_x to a_z by only one continuous projected harmonic zone $\chi_{I_1 \to I_2}^{\{a_x, a_{x+1}, \ldots, a_z\}} = [a_x I_1^s, min\{a_z I_1^e, I_2^e\}]$. By this combination of multipliers we can reduce the number of outgoing edges and vertices of the graph.

Algorithm 1 shows pseudo-code of the graph generation

process. It starts from task τ_1 and follows a depth first search structure. As described before, for each task τ_i, it constructs the list of projected harmonic zones which can be produced from the current interval $I_{v_{i,j}}$ and period range of the task. Then the resulting zones will be added to the tree and the search continues up to the last task, i.e., τ_n. To maintain correct indices for the vertices, we have assumed there is value K_i which will be incremented each time a vertex is added to the vertices related to task τ_i.

When the graph is constructed, each path that passes through n vertices, verifies existence of a harmonic relation between those potential sub-intervals defined in the vertices. For such path, the resulting sequence of intervals produce solution $I : \{I_1, I_2, \ldots, I_n\}$ with intervals I_i, $1 \leq i \leq n$.

Algorithm 1 Graph construction algorithm (GCA)

Input $\tau, G, I_{v_{i,j}}$ ▷ $I_{v_{i,j}}$ is the latest interval added to G
Output G

if $i = n$ **or** $T^{max}_{i+1} < I^s_{v_{i,j}}$ **then**
  return G
else
  $s \leftarrow max\{T^{min}_{i+1}, I^s_{v_{i,j}}\}$ ▷ update start point of T^{min}_{i+1}
  $a_1 \leftarrow \lfloor \frac{T^{max}_{i+1}}{I^e_{v_{i,j}}} \rfloor$
  $a_z \leftarrow \lfloor \frac{s}{I^s_{v_{i,j}}} \rfloor$
  $a_x \leftarrow \lfloor \frac{I^s_{v_{i,j}}}{I^e_{v_{i,j}} - I^s_{v_{i,j}}} \rfloor$ ▷ zones are continuous after a_x
  $m \leftarrow a_x - a_1$ ▷ number of disjoint zones
  $a \leftarrow a_1$
  for $k = 1$ **to** m **do**
    $I_{v_{i,k}} \leftarrow [max\{aI^s_{v_{i,j}}, s\}, min\{aI^e_{v_{i,j}}, T^{max}_{i+1}\}]$
    $q = k + K_i$ ▷ q is the latest index
    add vertex $v_{i+1,q}$
    add an edge between $v_{i,j}$ and $v_{i+1,q}$ to G
    GCA$(\tau, G, I_{v_{i,q}})$
    $a \leftarrow a + 1$
  end for
  $I_{v_{i,m+1}} \leftarrow [max\{aI^s_{v_{i,j}}, s\}, min\{a_z I^e_{v_{i,j}}, T^{max}_{i+1}\}]$
  $q = m + 1 + K_i$
  add vertex $v_{i+1,q}$
  add an edge from $v_{i,j}$ to $v_{i+1,q}$ to G
  GCA$(\tau, G, I_{v_{i,q}})$
end if
return G

Since projected harmonic zones are created from sorted intervals by the starting points, we never have a loop and the resulting graph G will be a tree. However, because of the fact that the number of harmonic multipliers highly depends on the period ranges, the number of vertices can exponentially grow. For example, even if there is only two disjoint set of harmonic zones between each two period ranges, i.e., where $\|A_{\tau_i \to \tau_{i+1}}\| = 2$, in the worst-case the number of vertices of the graph would be $O(2^n)$ which is exponential in the number of tasks.

As shown before, even when there are only two disjoint projected harmonic zones between two tasks, the complexity of the problem can be exponential. However, if tasks have tight harmonic relations, the problem can be solved in linear-time. The reason is that Algorithm 1 constructs only one projected harmonic zone in the period range of τ_i. It leads to considerable reductions in the size of the graph, i.e., the graph will have at most n number of vertices and $n-1$ number of edges. It leads to linear-time complexity in graph construction, as well as finding a path that passes through all vertices. Also verification of existence of tight harmonic relation between pairs τ_i and τ_{i+1} of tasks can be done in $O(n)$ since any of the conditions of Theorem 2 can be verified in $O(1)$.

There is another case where the problem can be solved in linear-time. It happens if for each two tasks τ_i and τ_{i+1}, $\|A_{\tau_i \to \tau_{i+1}}\| = 1$. The reason is that with one multiplier, only one projected harmonic zone between two tasks can be created.

However, if a task set has a combination of tight harmonic tasks (from conditions (3) or (4)), and tasks with single multipliers, i.e., $\|A_{\tau_i \to \tau_{i+1}}\| = 1$ (while (6) is not valid), still the problem can have exponential complexity. Assume we have three intervals $I_1 : [50, 60]$, $I_2 : [90, 110]$, and $I_3 : [500, 1500]$, thus because of (4), I_2 and I_3 have tight harmonic relation. However $\chi_{I_1 \to I_2} = [100, 110] \neq I_2$. Now we continue with $\chi_{[100,110] \to I_3}$. Since condition (4) is not valid any more, instead of one continuous projected harmonic zone we will have disjoint intervals $\chi_{[100,110] \to I_3} = \big\{[500, 550], [600, 660], \ldots, [900, 990], [1000, 1500]\big\}$. As it can be seen, if this situation happens in the graph, again the number of vertices grows rapidly.

Previously with the study of Cai et al., [22], it has been shown that having harmonic periods with period ratio $k_i = T_{i+1}/T_i$ equal to 2, or greater than 2, it is possible to find feasible non-preemptive schedule for the task set. Non-preemptive execution decreases delays between the sampling and actuation of the control tasks [20], [23] and simplifies the analysis of WCET of the tasks [24].

To construct customized harmonic relations, we can grow the graph by the multipliers which are favored by the designer. For example if the goal is to find a set of harmonic tasks with period ratio 2, the only step which can be done in Algorithm GCA is to restrict the unwanted multipliers.

V. Period Assignment to Guarantee Feasibility

A. Utilization Bounds of a Group of Tasks

In this section we show how we can calculate lower bound and upper bound of the utilization of the task set τ given a set of sub-intervals $I : I_1, I_2, \ldots, I_n$ produced by the algorithm in Sect. III. To calculate these bounds, we start by last task τ_n. According to the definition of projected harmonic zones, we guaranteed that any number in a resulting zone can be produced by a number in the source interval, that is why to find the solution, we go backward and start with the latest task. Once T_n is selected from I_n, the choices for T_{n-1} will be limited because not all values of the I_{n-1} are in harmonic relation with the specified T_n. To find the set of distinct points which have such relation with T_n, we use Definition 2. Accordingly, we have $b^1_{n-1} = \lfloor \frac{T_n}{I^e_{n-1}} \rfloor + 1$ as the smallest multiplier and $b^z_{n-1} = \lfloor \frac{T_n}{I^s_{n-1}} \rfloor$ as the largest multiplier. These two multipliers limit the possible values of

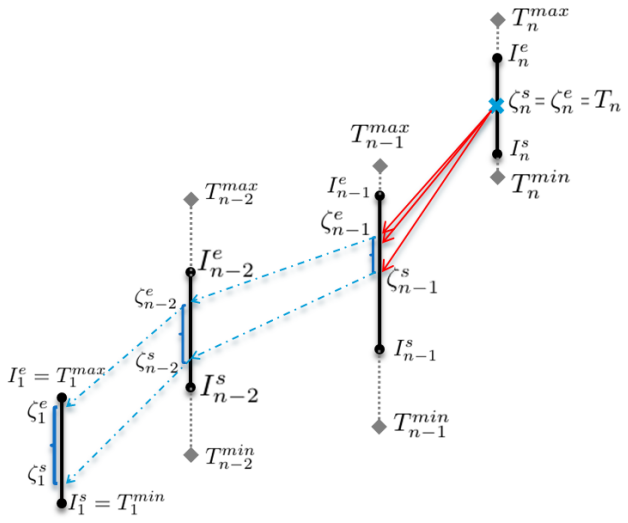

Fig. 5. A symbolic example to find utilization range of four tasks

T_{n-1} to the following continuous range

$$\zeta_{n-1} : \left[\frac{T_n}{\lfloor \frac{T_n}{I^s_{n-1}} \rfloor}, \frac{T_n}{\lfloor \frac{T_n}{I^e_{n-1}} \rfloor + 1} \right] \tag{9}$$

where ζ_{n-1} shows the valid range of T_{n-1}. In the next step we find the bounds of T_{n-2}. The smallest and the largest multipliers that can be applied on I_{n-2} to produce a number in I_{n-1} are denoted by $a_1 = \lfloor \frac{I^s_{n-2}}{I^e_{n-1}} \rfloor + 1$ and $a_z = \lfloor \frac{I^e_{n-1}}{I^s_{n-2}} \rfloor$ respectively. Note that both of them are defined in Definition 2. Now for task τ_{n-2} we have.

$$\zeta_{n-2} : \left[max\left\{ \frac{\zeta^s_{n-1}}{\lfloor \frac{I^e_{n-1}}{I^s_{n-2}} \rfloor}, I^s_{n-2} \right\}, min\left\{ \frac{\zeta^e_{n-1}}{\lfloor \frac{I^s_{n-2}}{I^e_{n-1}} \rfloor + 1}, I^e_{n-2} \right\} \right] \tag{10}$$

Fig. 5 shows a symbolic example of ζ_{n-2}. With the same method we can extend (10) to task τ_i as

$$\zeta_i : \left[max\left\{ \frac{\zeta^s_{i+1}}{\lfloor \frac{I^e_{i+1}}{I^s_i} \rfloor}, I^s_i \right\}, min\left\{ \frac{\zeta^e_{i+1}}{\lfloor \frac{I^s_i}{I^e_{i+1}} \rfloor}, I^e_i \right\} \right] \tag{11}$$

Using (9) and (11), for each task τ_i, $1 \leq i \leq n-1$ we will have a period bound which can be used to calculate the minimum and maximum possible utilization of the task set. Maximum utilization is obtained when the shortest period, i.e., ζ^s_i is used and minimum utilization is obtained when ζ^e_i is considered as task period. Thus utilization of task τ_i will be larger than $\frac{c_i}{\zeta^e_i}$ and will be smaller than $\frac{c_i}{\zeta^s_i}$. Finally, system utilization is bounded by:

$$U^{low} = \sum_{i=1}^{n} \frac{c_i}{\zeta^e_i} \tag{12}$$

$$U^{high} = \sum_{i=1}^{n} \frac{c_i}{\zeta^s_i} \tag{13}$$

Formulas (12) and (13) have linear-time computational complexity since calculation of ζ_i can be done in $O(1)$. According to (11), ζ_i depends on the solution ranges of other tasks with larger I^s_j, $i < j \leq n$, values, and T_n.

B. Period Assignment Algorithm

Period assignment can be done with different goals, e.g., having high or low system utilization. But the main common goal is to keep the system feasible. As it has been shown in the previous subsection (Sect. V-A), when one period is selected for T_n, τ_{n-1} will have several choices because of multipliers $b^1_{n-1} = \lfloor \frac{T_n}{I^e_{n-1}} \rfloor + 1$ to $b^z_{n-1} = \lfloor \frac{T_n}{I^s_{n-1}} \rfloor$. As a result, it is not easy to find optimal period values such that the task set is harmonic and utilization remains below 1. However in this subsection, we present a greedy algorithm to find low and high utilization task sets.

As said before, the largest or the smallest values of T_{n-1} are obtained using b^1_{n-1} or b^z_{n-1}, respectively. Based on this idea, first we introduce Algorithm 2 (or LU for short) which takes resulting path I as input and produces set $T = \{T_1, T_2, \ldots, T_n\}$ of periods. The goal of this algorithm is to minimize the utilization. It starts with task τ_n and set T_n to I^e_n. Then it calculates $\lfloor \frac{T_n}{I^e_{n-1}} \rfloor$ which is the smallest divisor that maps T_n to I_{n-1}. When a small divisor is chosen, the resulting value will be the largest possible candidate that has a harmonic relation with T_n. In fact, the algorithm looks for the largest period for the next task in each step.

Algorithm 2 Low utilization period assignment algorithm

Input I ▷ I is the list of ranges with harmonic relations
Output T ▷ T is the set of final periods

1: $T_n \leftarrow I^e_n$
2: **for** $i \leftarrow n-1$ **downto** 1 **do**
3: $b_i \leftarrow \lfloor \frac{T_{i+1}}{I^e_i} \rfloor$
4: $T_i \leftarrow T_{i+1} \div b_i$
5: **end for**
6: **return** T

If Line 1 of Algorithm 2 is replaced by $T_n \leftarrow I^s_n$, and Line 3 is replaced by $b_i \leftarrow \lfloor \frac{T_{i+1}}{I^s_i} \rfloor$, the algorithm produces a task set with high utilization because it starts with the smallest possible value for T_n and looks for the biggest divisor. We call this algorithm as HU.

In the experiments we have implemented both LU and HU algorithms as well as an optimal period assignment algorithm that searches for the task set with the highest feasible utilization, i.e., the resulting task set will have the shortest periods which is a favorable property in many applications such as control systems [20]. The optimal algorithm starts with $T_n \leftarrow \lceil I^s_n \rceil$ and gradually increments T_n (by 1) to reach $\lfloor I^e_n \rfloor$. In each step it uses an exhaustive recursive search over all divisors of obtained period values to find a period assignment with the highest feasible utilization. Computational complexity of this algorithm is equal with the complexity of an exhaustive search on our graph multiplied by the length of the period range of τ_n, hence, in practice, this algorithm might not be an applicable solution.

It is worth mentioning that ordering of the tasks affects the resulting utilization of period assignment algorithms, however, because of practical limitations, we could not consider all

possible orderings of the tasks in calculation of optimal period values. The last step of period assignment algorithm is to find closest integer period to the obtained period value of LU and HU algorithms. If we assume periods are not very short, then the rounding of the results can have negligible effects on the utilizations. One way to cope with the problem is to scale period before they are given to the model. The further effects of rounding the periods and quantifying its error will remain as one of our future works.

VI. Experimental Results

In this section we compare the performance of the Algorithm 1 and Sr algorithm developed by [4]. When a solution is found by Algorithm 1, we use LU, HU, and Optimal period assignment methods described in Sect. V-B to assign a period to the tasks. Performance measures are acceptance ratio (AR), i.e., the ratio of harmonic task sets with feasible utilization to the total number of generated task sets for each run, as well as the utilization of the accepted task sets. The simulations have been performed using the discrete event simulator (DEVS-Suite) [25].

Since to the best of our knowledge, there exists no algorithm to produce harmonic task sets from period ranges, we modified the Sr algorithm [4] to cope with period ranges. Originally, Sr maps given period values of each task τ_i, to $r \times 2^{\lfloor log(T_i/r) \rfloor}$ where r is a base value in range $(0.5T_1, T1]$. For example if $r = 3$, tasks with periods 4 and 5 will be mapped to r while a task with period 13 will be mapped to 12 which is 4 times r. However since we have period ranges and we do not know T_i beforehand, we assign T_i to the smallest value with pattern $r2^x$ which is inside the range $[T_i^{min}, T_i^{max}]$. Note that x is an integer number smaller than $\lfloor log(max\{T_i^{max}\}_{\forall i}/r) \rfloor$. Since the choice for r depends on T_1 which we do not have, first we assume $T_1 = T_1^{min}$ then we gradually increase it to T_1^{max} and we count the number of successfully mapped period values (accepted tasks) in each step. Finally, a solution with maximum number of accepted tasks will be reported as output of the algorithm.

In the next two subsections, we generate random tasks according to [5] workload model as well as random tight harmonic harmonic task sets.

A. Random Tasks with Workload Model of [5]

In [5], period ranges are generated randomly as $T_i^{max} = rand[100, 5000]$ and $T_i^{min} = T_i^{max}(1 - \sigma)$ where $\sigma \in [0.3, 0.8]$ is the *tolerance* parameter and shows the wideness of period ranges with respect to their ending point. In our first experiment, we consider 10 tasks, then for each σ value we repeat the experiments 1000 times and calculate average values of AR and utilization. For confidence level of 0.95, confidence interval of the experiment was 0.05 for each measure. Also to create random execution times, first we have generated random utilization values by uUniFast method [26] for utilization 1. Then we have multiplied each utilization u_i by T_i^{min} obtained in the previous step. In this way, any period assignment will result in feasible utilization since even if tasks are executed by their shortest period, still the utilization will remain less than 1.

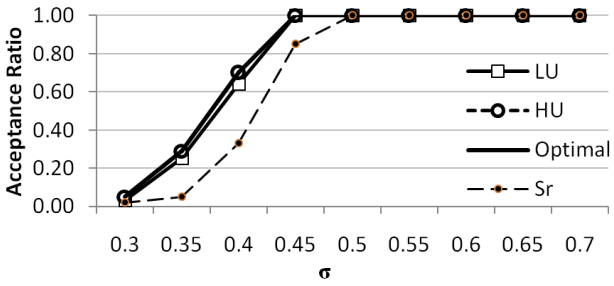

Fig. 6. Acceptance ratio of the algorithms with workload model [5]

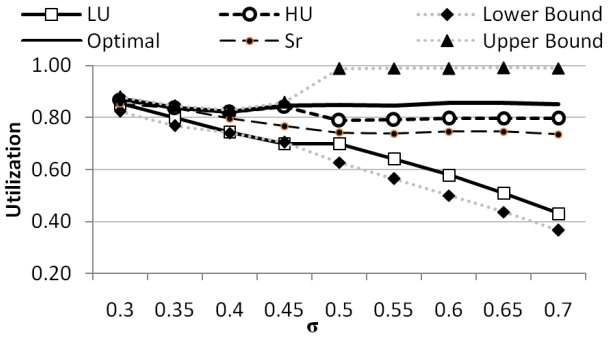

Fig. 7. Utilization of the algorithms with workload model [5]

Fig. 6 and Fig. 7 illustrate AR and utilization of the algorithms for different tolerance values (σ in the horizontal axis). As shown in Fig. 6, for σ values smaller than 0.3, none of the algorithms produce a harmonic task set. The AR of the algorithms increases with increasing σ, because when the period ranges become wide (for large σ values), they have more intersections with each other. This produces more chances for the existence of integer multipliers between ranges. Thus, even the Sr algorithm finds many harmonic task sets. On the other hand, when σ is more than 0.5, more harmonic task sets are found. Consequently, the LU algorithm has more chances to look for task sets with low utilizations as shown in Fig. 7. The resulting utilization value of Sr is rather high especially for $\sigma > 0.5$. The reason is that the smallest integer number x which can satisfy $r \times 2^x \in [T_i^{min}, T_i^{max}]$ will be the task period in our implementation of Sr. Finally, for tasks with wide period ranges, HU performs nearly as good as the optimal method although it is a linear-time greedy algorithm. Fig. 7 also shows utilization bounds obtained from formula (11). When σ is smaller than 0.45, both bounds are close to each other which shows that period assignment methods have not so much choices in the final utilization. Although the lower bound decreases with increasing σ, it is still similar to the utilization achieved by LU. This shows that LU has good performance in period assignment when the goal is to find low utilization task sets. The upper bound of the utilization remains 1 because of the assumptions of this experiment. Later we will discuss cases where the utilization can go beyond one, and then we evaluate the performance of the period assignment algorithms.

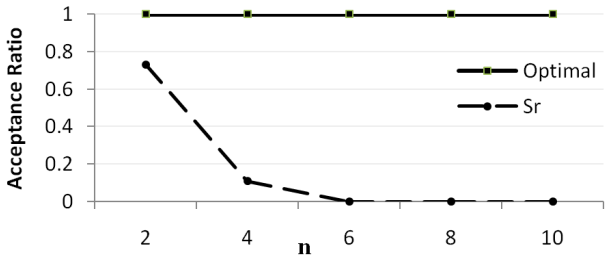

Fig. 8. Acceptance ratio of the algorithms for different number of tasks (Optimal and LU are overlapping)

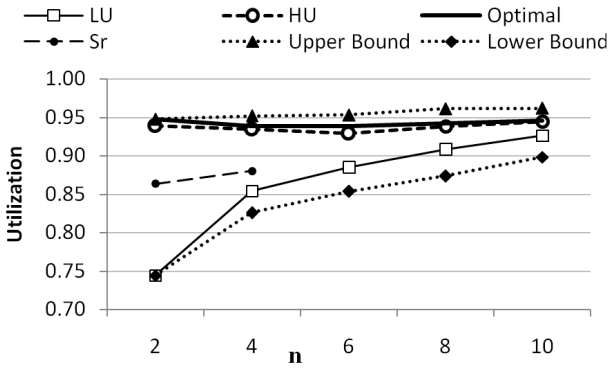

Fig. 9. Utilization of the algorithms for different number of tasks.

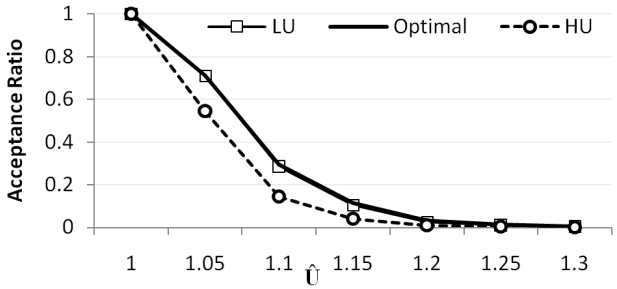

Fig. 10. Acceptance ratio of the algorithms for workloads with $\tilde{U} \geq 1$

B. Random Tasks with Tight Harmonic Relations

In this subsection, we consider task sets with tight harmonic relations. To generate those task sets, for each pair of consecutive tasks we use one of the conditions (4), (5), or (6) with a uniform chance. In the first case, we calculate p based on (4) according to the previous interval which is $[T_{i-1}^{min}, T_{i-1}^{max}]$. Then the period range of τ_i is generated as $[p, (1+x)p]$, where x is randomly selected from $[0, \sigma]$ for tolerance value $\sigma = 0.4$. For two cases where conditions (5) or (6) have to be satisfied, first we generate one random integer value $k \in \{1, 2, 3, \ldots, 5\}$ as the multiplier. Next we generate two random values x and y in range $[kT_{i-1}^{min}, kT_{i-1}^{max}]$. Then we assign the smaller value to T_i^{min} and the larger value to T_i^{max}. If $k = 1$, the next period range fits into the current period range, otherwise, condition (5) will hold. When period ranges are produced, we assign execution times of the tasks as $c_i = u_i \tilde{U} T_i^{min}$, where u_i is a random utilization value for task τ_i obtained from the uUniFast algorithm (considering system utilization 1) [26], and $\tilde{U}$ is the potential system utilization. By multiplying u_i with $\tilde{U}$ we generate cases where the system utilization will be more than 1 if all tasks are executed by their T_i^{min}. In other words, using $\tilde{U}$ we can evaluate the performance of the period assignment algorithms because then the choice of period becomes important in the feasibility of the task set.

In the first experiment we consider different values of n from 2 to 10, and $\tilde{U} = 1$. The reason for this limited n is that the Sr algorithm was not able to find any harmonic task set when there were more than 6 tasks in the system. To obtain results, we have repeated each setup value 1000 times, and the results were in confidence interval 0.05 for confidence level 0.95. Fig. 8 and Fig. 9 show the AR and utilization of the algorithms for different values of n (in the horizontal axis). Note that since Sr did not produce any harmonic task set after $n \geq 6$, there was no data to report. As shown in Fig. 8, AR of Sr algorithm is highly sensitive to the number of tasks in the system and drops rapidly with increasing n, even though in this experiment, tasks are tight harmonic and there are many chances to find harmonic task sets.

Moreover, Fig. 9 shows that HU is relatively close to the optimal algorithm (and starting with $n = 5$ results overlap). Besides, when the number of tasks increase, utilization of HU, LU, and the optimal algorithm becomes close to each other because when n increases, the resulting solution from Algorithm 1 narrows, which lead to less choices for the period assignment algorithms. For the same reason, utilization bounds also get tighter with the increase in n. Finally, it is interesting that lower and upper bounds of the utilization calculated by (11) are very close to the results of the LU and HU algorithms, respectively. It means that these bounds are good estimations for an achievable utilization.

To compare the performance of the period assignment algorithms, in the next experiment we consider $\tilde{U}$ from 1.0 to 1.5 and $n = 10$. Fig. 10 and Fig. 11 show AR and utilization of the algorithms for different values of $\tilde{U}$ (the horizontal axis). As shown in Fig. 10, when $\tilde{U}$ is greater than 1.15, the number of feasible task sets reduces rapidly even for the optimal algorithm. HU has a lower rate of acceptance because it tries to find task sets with higher utilizations which might be infeasible in this experiment. Moreover, AR of LU is very close to the optimal algorithm which means that whenever $\tilde{U}$ is greater than 1, LU is an efficient period assignment algorithm.

As shown in Fig. 11, the upper bound of the utilization grows with the growth of $\tilde{U}$ up to 1.25 at the end. It means that although $\sum_{i=1}^{n}(c_i/T_i^{min})$ is equal to $\tilde{U}$, the upper bound of the utilization which is obtained by formula (11) is tighter than simply considering $\tilde{U}$ as the upper bound. Having the lower bound very close to LU, it is possible to deduce that the sequence of intervals with harmonic relations with each other which are obtained from Algorithm 1, were finally narrow. That is why LU has not so much chances to decrease the utilization, and consequently, the lower bound is close to LU.

VII. Conclusion

In this paper, we have presented a model to describe harmonic relations between ranges of period values, rather than single numbers only. The resulting harmonic task sets can be

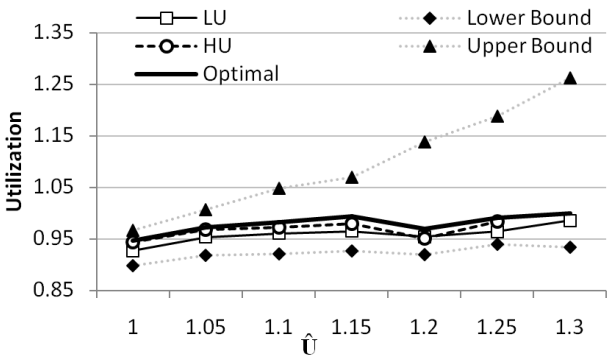

Fig. 11. Utilization of the algorithms for workloads with $\tilde{U} \geq 1$

scheduled with fixed priority scheduling algorithms such as RM while their periodic behavior is guaranteed.

Our method identifies sub-intervals within given period ranges which are in harmonic relations with each other. Any number inside the resulting sub-interval of a task can be generated from the previous task with smaller period range. Based on the existence of harmonic relations between period ranges, we have introduced a graph to represent the task set. Using this graph, it is possible to construct customized harmonic relations by restricting non-favorable multipliers.

We derived sufficient conditions for the existence of a linear-time harmonic task set as well as utilization bounds of each resulting harmonic range. They provide more flexibility for the designers to have system with high or low utilizations. Results from a simulation study underline the effectiveness of the approach. As the extensions of this work, we tackle the problem of ordering of the tasks which affects on the utilization of the final solution. We also will dealt with the effect of period rounding for the applications with discrete period values.

ACKNOWLEDGMENT

The authors would like to thank Thomas Fehmel and Stefan Schorr for their helpful discussions. We are grateful to the anonymous reviewers for many valuable comments. The work presented in this paper has been supported by the German Academic Exchange Service (DAAD) scholarship.

REFERENCES

[1] Y. Wu, G. Buttazzo, E. Bini, and A. Cervin, "Parameter selection for real-time controllers in resource-constrained systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 610–620, 2010.

[2] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[3] F. Eisenbrand and T. Rothvoß, "EDF-schedulability of synchronous periodic task systems is coNP-hard," in *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2010, pp. 1029–1034.

[4] C.-C. Han and H.-Y. Tyan, "A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms," in *IEEE Real-Time Systems Symposium (RTSS)*. IEEE Computer Society, 1997, pp. 36–45.

[5] I. Ripoll and R. Ballester-Ripoll, "Period selection for minimal hyperperiod in periodic task systems," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1813–1822, 2013.

[6] J. Goossens and C. Macq, "Limitation of the hyper-period in real-time periodic task set generation," in *RTS Embedded System (RTS)*, 2001, pp. 133–147.

[7] 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 *International Workshop on Real-Time Computing Systems and Applications (RTCSA)*, 1996, pp. 195–202.

[8] 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 (RTAS)*, 2003, pp. 10–18.

[9] 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 (RTSS)*, 2003, pp. 210–219.

[10] S. Anssi, S. Kuntz, S. Gérard, and F. Terrier, "On the gap between schedulability tests and an automotive task model," *Journal of Systems Architecture*, vol. 59, no. 6, pp. 341–350, 2013.

[11] 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 (RTAS)*, 2010, pp. 111–120.

[12] V. Bonifaci, A. Marchetti-Spaccamela, N. Megow, and A. Wiese, "Polynomial-time exact schedulability tests for harmonic real-time tasks," in *IEEE Real-Time Systems Symposium, (RTSS)*, 2013.

[13] V. Brocal, P. Balbastre, R. Ballester, and I. Ripoll, "Task period selection to minimize hyperperiod," in *IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, 2011, pp. 1–4.

[14] A. Crespo, I. Ripoll, and M. Masmano, "Partitioned embedded architecture based on hypervisor: The xtratum approach," in *Dependable Computing Conference (EDCC)*, 2010, pp. 67–72.

[15] H. Kopetz, "On the design of distributed time-triggered embedded systems," *Journal of Computing Science and Engineering*, vol. 2, no. 4, pp. 340–356, 2008.

[16] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 289–302, 2002.

[17] N. Min-Allah, I. Ali, J. Xing, and Y. Wang, "Utilization bound for periodic task set with composite deadline," *Computers and Electrical Engineering*, vol. 36, no. 6, pp. 1101–1109, 2010.

[18] T.-W. Kuo and A. Mok, "Load adjustment in adaptive real-time systems," in *IEEE Real-Time Systems Symposium (RTSS)*, 1991, pp. 160–170.

[19] M. Fan and G. Quan, "Harmonic semi-partitioned scheduling for fixed-priority real-time tasks on multi-core platform," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2012, pp. 503–508.

[20] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *IEEE Real-Time Systems Symposium (RTSS)*, 2008, pp. 291–300.

[21] J. Theis and G. Fohler, "Transformation of sporadic tasks for off-line scheduling with utilization and response time trade-offs," in *Real-Time and Network Systems (RTNS)*, 2011, pp. 119–128.

[22] Y. Cai and M. C. Kong, "Nonpreemptive scheduling of periodic tasks in uni- and multiprocessor systems," *Algorithmica*, vol. 15, no. 6, pp. 572–599, 1996.

[23] M. Nasri and M. Kargahi, "A method for improving delay-sensitive accuracy in real-time embedded systems," in *IEEE Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2012, pp. 378–387.

[24] G. C. Buttazzo, M. Bertogna, and G. Yao, "Limited preemptive scheduling for real-time systems: A survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 3–15, 2013.

[25] "Discrete-event system simulator (DEVS-suite)," 2009. [Online]. Available: http://acims.asu.edu/software/devs-suite

[26] E. Bini and G. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.