An Efficient Method for Assigning Harmonic Periods to Hard Real-time Tasks with Period Ranges

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

***Abstract*—During the design phase of many real-time systems, designers often have a *range* of acceptable period values for which some levels of safety or quality of service are guaranteed. The choice of period values influences system schedulability and computational complexity of schedulability analysis, especially for the rate monotonic (RM) scheduling algorithm. It has been shown that RM guarantees 100% utilization if the periods are harmonic, i.e., each period is an integer multiple of shorter periods. In this paper, we address harmonic period assignment problem where each task has a given period range.**

We extend the results of our previous work and present an $O(n^2log(n))$ algorithm (where n is the number of tasks) to verify necessary and sufficient conditions for the existence of a harmonic period assignment in cases where the previous solution has pseudo-polynomial computational complexity. We provide utilization bounds of the potential assignments as well as a heuristic algorithm to construct low utilization harmonic task sets. The efficiency of our period assignment algorithms has been evaluated in terms of acceptance ratio, task set utilization, data structure size, and the number of operations required for harmonic period assignment.

I. Introduction

Many real-time applications have recurrent behavior which can be described by periodic tasks with implicit deadlines. In control systems, for example, the stability of the control tasks is maintained [1], as long as the periodic behavior is guaranteed, In these systems, periods are not limited to single values, rather, *ranges* of acceptable period values for each task are available to designers.

Period assignment is an important design step of real-time systems because periods are a bridge to relate the performance and safety of a system to its schedulability. For example, in control applications, quality of control is a function of the sampling periods; the shorter the period, the higher the quality of control [2]. However, short periods increase task set utilization which impedes schedulability. Period assignment is important because it affects the computational complexity of the schedulability tests as well as the worst-case response time (WCRT) analysis methods [3]. For example, even for highly utilized task sets, if the periods are harmonic, i.e., each period is an integer multiple of the shorter periods, it is possible to have a linear-time schedulability test [4]. Moreover, response time analysis of harmonic tasks can be done in polynomial-time [5], [6].

Harmonic periods have been employed in a large spectrum of industrial real-time applications such as avionics, submarines, and robotics [7], [8], [9], [10], [11]. These tasks can be scheduled by RM up to 100% utilization [4]. Also, the hyperperiods of such task sets are relatively small [12], [13] and it is equal to the largest period, which is a major concern for hypervisors in virtualization [14] or time triggered systems [15] that use offline scheduling tables. Moreover, fault tolerance mechanisms for fault detection and recovery can be efficiently implemented if periods of the tasks are harmonic [16]. In this paper, we study the problem of harmonic period assignment in real-time systems in which each individual task has a *range* of valid periods.

In our previous work [17], we have provided a framework to construct customized harmonic periods from a set of given period ranges. The solution is based on finding all possible projected harmonic zones, i.e., sub-intervals of a target interval which can be in a harmonic relation with a source interval. In this solution, called *forward search* hereafter, starting from the first period range, we construct a graph of projected harmonic zones between all period ranges using depth-first search. In this graph, every connecting path from the first period range to the last one, is a potential set of intervals that guarantees the existence of a harmonic period assignment. However, the size of this graph might be exponential w.r.t the number of tasks because it depends on the number of integer multipliers between two intervals. In that work, we have also derived sufficient conditions for an optimal linear-time solution.

In this paper we present a new method to address those cases which are sources of the exponential growth in our previous forward approach. The main idea is to use a *backward search* from the period range of the last task to the first one, and extract and store the *effective origins* of potential harmonic relations. We show that our new approach is efficient especially when there are a lot of disjoint integer multipliers between two intervals. We show that if non-intersecting projections exist from a source to a destination interval, the harmonic period assignment problem can be solved in $O(n^2log(n))$ where n is the number of tasks. The key to reduce computational complexity is to avoid keeping track of all integer multipliers; we only focus on the multipliers that produce distinct projected harmonic zones, and we show if more than two of them exist between two intervals, there is no need to consider the remaining integer multipliers.

Using this idea, we provide a test to verify necessary and sufficient conditions for the existence of a harmonic period assignment, and show that this test is valid even if the given intervals are overlapping. In other words, there is no need to consider different task orderings for our test. Finally, as a tool for system designers, we provide utilization bounds of the

1068-3070/15 $31.00 © 2015 IEEE
DOI 10.1109/ECRTS.2015.21

potential solutions together with a heuristic algorithm which constructs low utilization task sets with harmonic periods.

The remainder of the paper is organized as follows; related work is presented in Sect. II which is followed by the system model in Sect. III. In Sect. IV, we introduce our new existence test for harmonic period assignment. Sect. V derives utilization bounds of the potential period assignments and presents a heuristic algorithm to assign harmonic periods. Sect. VI discusses the theoretical achievements and open problems. Effectiveness of the proposed solution is evaluated in Sect. VII, and finally the paper is concluded in Sect. VIII.

II. Related Work

Interesting properties of harmonic tasks have motivated several works to convert periodic task sets to harmonic ones [4], [9], [18], [17]. For instance, to provide a schedulability test for RM, Sr and DCT algorithms have been introduced in [4] to assign artificial harmonic periods which are smaller than the original ones. Accordingly, the original task set is schedulable if such an assignment exists and the resulting utilization is smaller than 1. In the Sr algorithm, the ratio of two consecutive periods is a power of 2, and in DCT it can be an arbitrary integer value. Also to verify schedulability of a system with deadlines larger than periods, in [18], harmonic deadlines have been created using the least common multiplier of the previous periods. However, in these studies, period ranges are not considered. Also tasks are executed with their original non-harmonic periods.

Later in [9], Sr and DCT algorithms [4] have been applied to construct harmonic periods for radar systems. In [19], Sr is used to build partitions of tasks with harmonic periods for multiprocessor systems. This harmonic partitioning problem has been previously tackled by Kue et al. [20] who have introduced a method to obtain a minimum number of harmonic chains, i.e., subsets of tasks with harmonic periods. However, none of these works consider ranges of periods.

To minimize the hyperperiod of a task set described with period ranges, Ripoll et al. [13] presented an algorithm to obtain the minimum hyperperiod by finding the first intersection between integer multiples of all period ranges. However, this approach cannot be used to find a harmonic period assignment since it does not force harmonic relation between the final periods. Moreover, computational complexity of this approach is pseudo-polynomial which may limit its applicability.

In our previous work [17], we have provided a framework to construct customized harmonic periods from given period ranges. The solution is based on finding all possible projected harmonic zones between the source interval and destination interval (note that intervals are indexed by their starting values). Every value in such a zone is an integer multiple of a value in the source interval. As shown by Fig. 1, we represent the relation between those zones with a graph where edges are integer multipliers and vertices are the projected zones. To build the graph, we use a depth first search from the first interval to the last interval. This approach is called forward search because at each expansion, it finds all projected harmonic zones between the current vertex and the next interval, and merges the overlapping projections into one vertex. At the end, any path that connects the first vertex to one of the vertices of the last task is a potential solution which guarantees the existence of a harmonic period assignment. However, as it can be seen in Fig. 1, the size of the graph may grow exponentially since it depends on the number of *disjoint* projected harmonic zones between two intervals. In [17] we have shown that if all of the projected zones are overlapping (defined as tight harmonic relation), the problem can be solved in $O(n)$ because the graph will only have one vertex per task. In this paper, we use a backward approach (from the last task to the first one) to show that in the cases where the projected harmonic zones are disjoint, the problem can be solved in $O(n^2 log(n))$ where n is the number of the tasks.

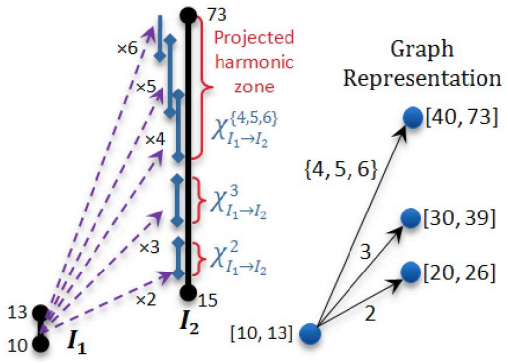

Fig. 1. Graph of harmonic relations in the forward approach [17]. The overlapping projected harmonic zones are merged and represented as one vertex in the graph.

III. System Model and Problem Statement

We assume a uni-processor system with a hard real-time task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ with n independent periodic tasks. Each task τ_i is described by $\tau_i : (c_i, T_i^{min}, T_i^{max})$, where c_i is the worst-case execution time, and $T_i^{min}, T_i^{max} \in \mathbb{N}$ are the minimum and maximum values for the task's period denoted by $T_i \in \mathbb{R}$. Later in Sect. IV-C we discuss integer value periods. We assume that tasks are indexed as $T_1^{min} \leq T_2^{min} \leq \ldots \leq T_n^{min}$ and have no release-offset.

The goal of the paper is to assign period values $T_i \in [T_i^{min}, T_i^{max}]$ for each task τ_i (for $1 \leq i \leq n$) such that all periods of the tasks become harmonic, i.e., $T_j = k_j \times T_{j-1}$ for $1 < j \leq n$ where $k_j \in \mathbb{N}$ is the period ratio of two consecutive tasks and an integer number greater than or equal 1.

We aim to find periods such that the system utilization is smaller than or equal 1, i.e., $U \leq 1$ where $U = \sum_{\tau_i \in \tau} u_i$ and for each task τ_i, we have $u_i = c_i/T_i$. Note that in this paper, $\mathbb{N} = \{1, 2, 3, \ldots\}$. We denote an interval by $I = [I^s, I^e]$ where I^s and I^e represent start and end of the interval, respectively.

IV. Existence Test for Harmonic Period Assignment

In this section we present a test to verify the existence of a harmonic period assignment. We show that our test is valid even if period ranges have intersections. Finally, we extend the test to the cases where periods must be integer numbers.

We start by some important definitions from our previous work [17] on constructing harmonic periods for the tasks.

Definition 1. Projected harmonic zone $\chi^a_{I_1 \to I_2} : [\chi^s, \chi^e]$ from interval I_1 to I_2 ($I^s_1 \leq I^s_2$), with multiplier $a \in \mathbb{N}$, is a range of numbers in I_2 from $\chi^s = max\{I^s_2, aI^s_1\}$ to $\chi^e = min\{I^e_2, aI^e_1\}$, and for any $i_2 \in I_2$ there exists at least one $i_1 \in I_1$ such that $i_2/i_1 \in \mathbb{N}$.

In [17], we have shown how to calculate all possible integer multipliers between two intervals I_1 and I_2 ($I^s_1 \leq I^s_2$). Using these multipliers we derive harmonic relations between I_1 and I_2, denoted by $X_{I_1 \to I_2} \in I_2$, as a set of projected zones from all existing multipliers between I_1 and I_2. Fig. 1 illustrates an example of $X_{I_1 \to I_2}$. Consequently, for any value $i_2 \in X_{I_1 \to I_2}$ there exists at least one $i_1 \in I_1$ so that $i_2/i_1 \in \mathbb{N}$.

Finally, according to [17], the existence of a harmonic period assignment between a set of intervals is defined as

Definition 2. For the set of intervals $I_1, I_2, \ldots, I_n$, a harmonic period assignment exists if there exist values $i_1 \in I_1, i_2 \in I_2, \ldots, i_n \in I_n$ such that $\frac{i_2}{i_1} \in \mathbb{N}$, $\frac{i_3}{i_2} \in \mathbb{N}$, $\ldots$, $\frac{i_n}{i_{n-1}} \in \mathbb{N}$.

A. Backward Solution Approach

As shown in our previous work [17], the existence of a harmonic relation between two intervals I_1 and I_2, and also between intervals I_2 and I_3, does not necessarily mean that a harmonic period assignment exists between I_1, I_2, and I_3. The non-existence of such an assignment is due to the fact that I_1 may not have any harmonic relation with the projected harmonic zones between I_2 and I_3. In other words, integer multiples of I_1 have no intersections with the projected harmonic zones from I_2 to I_3. We define an integer multiple of an interval as

Definition 3. Integer multiple of interval I with multiplier $a \in \mathbb{N}$ is shown by $\phi^a = [\phi^s, \phi^e]$ where $\phi^s = aI^s$ and $\phi^e = aI^e$. For any value $x \in \phi^a$, there exists a value $y \in I$ so that $\frac{x}{y} \in \mathbb{N}$.

As shown in Fig. 2, having an intersection between at least one integer multiple of any of the intervals with I_n is not sufficient to guarantee the existence of a harmonic period assignment because those multipliers might not necessarily be integer multiples of each other.

While in the forward approach [17], we build the graph (e.g. Fig. 1), from the first task to the last one, in the backward approach presented in this paper, we start from the last interval, i.e., I_n, and try to find sub-intervals of I_{n-1} which are the original source of intersecting integer multiples of I_{n-1} with I_n. We call these sub-intervals as *projection origins* of I_{n-1} from I_n.

Definition 4. Projection origins of interval I_i from interval I_{i+1} ($I^s_i \leq I^s_{i+1}$) are denoted by $\psi_i = \{\psi_{i,1}, \psi_{i,2}, \ldots, \psi_{i,h_i}\}$ where h_i is the number of sub-intervals inside ψ_i and we have

$$\forall x \in \psi_i, \exists y \in I_{i+1};\ \text{such that } \frac{y}{x} \in \mathbb{N} \tag{1}$$

$$\forall j, k \in \{1, 2, \ldots, h_i\}, j \neq k; \psi_{i,j} \cap \psi_{i,k} = \emptyset \tag{2}$$

$$\forall j, 1 \leq j \leq h_i : I^s_i \leq \psi^s_{i,j} \text{ and } \psi^e_{i,j} \leq I^e_i \tag{3}$$

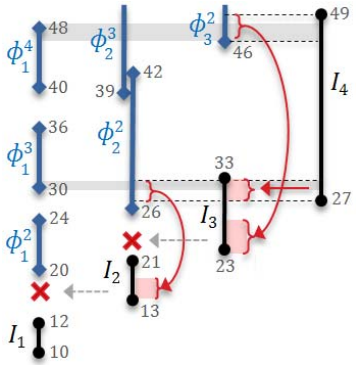

Fig. 2. An example to show that intersection between integer multiples of all intervals with I_n is not sufficient for the existence of a harmonic period assignment.

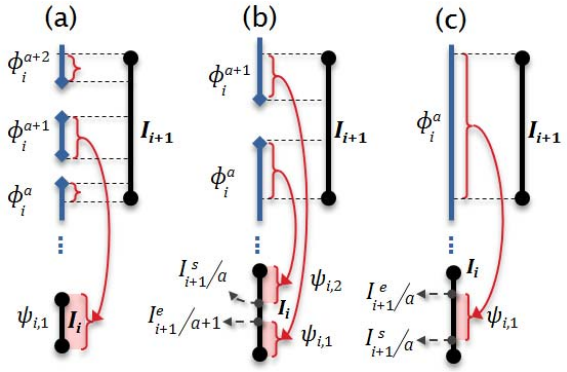

Fig. 3. Three cases for the projection origins of I_i from I_{i+1}; a) at least one of the integer multiples of I_i is totally inside I_{i+1}, b) integer multiples of I_i do not have complete intersection with I_{i+1}, and c) integer multiple of I_i is larger than I_{i+1} and it creates only one projection origin

Fig. 3 illustrates projection origins of I_i from I_{i+1}. The most important observation in this figure is that if there are more than 2 disjoint integer multiples between I_i and I_{i+1}, ψ_i will contain only one interval because in that case, the second integer multiple, shown by ϕ^{a+1}_i in Fig. 3-(a), will be fully covered by I_{i+1}. The following property summarizes this observation which is the key factor to reduce computational complexity of the solution.

Property 1. For two intervals I_i and I_{i+1}, the number of sub-intervals in the projection origins of I_i from I_{i+1} is smaller than or equal to 2, i.e., $h_i \leq 2$, if the overlapping integer multiples of I_i with I_{i+1} do not have intersection with each other, namely

$$\bigcap_{\forall a; \phi^a_i \cap I_{i+1} \neq \emptyset} \phi^a_i = \emptyset \tag{4}$$

Proof: First we show that if there are $b > 2$ disjoint integer multiples between I_i and I_{i+1}, h_i will be 1. As illustrated in Fig. 3-(a), if the number of disjoint integer multiples is greater than 2, according to Definition 4, all values

of ϕ_i^{a+1} (from $(a+1)I_i^s$ to $(a+1)I_i^e$), will intersect with I_{i+1}. This means that condition (1) holds for all values of I_i because we have $\forall x \in I_i, \exists y = (a+1)x, y \in I_{i+1}$ so that $x/y \in \mathbb{N}$. Also as shown by Fig. 3-(b), if $b \leq 2$, there will be at most two disjoint members in set ψ_i; one of them covers the starting values of I_i from I_i^s to $I_{i+1}^e/(a+1)$, and the other covers ending values of I_i from I_{i+1}^s/a to I_i^e. If $(a+1)I_i^s < aI_i^e$, which means that the integer multiples intersect, h_i reduces to 1 as these two sub-intervals cover all values of I_i. Note that we merge sub-intervals which intersect with each other because according to (2), ψ_i must partition the original interval I_i. ■

The following formulas derive the starting and ending integer multipliers a^s and a^e for which the integer multiples of I_i overlap with I_{i+1} (from Definition 2 in [17])

$$a^s = \left\lfloor \frac{I_{i+1}^s}{I_i^e} \right\rfloor + \begin{cases} 0, & \frac{I_{i+1}^s}{I_i^e} \in \mathbb{N} \\ 1, & \text{otherwise} \end{cases} \tag{5}$$

$$a^e = \left\lfloor \frac{I_{i+1}^e}{I_i^s} \right\rfloor \tag{6}$$

In our previous work [17], we have derived a necessary condition to have integer multiples of I_i which are overlapping with I_{i+1}. Using this formula we obtain conditions for which Property 1 is valid for two intervals I_i and I_{i+1} as

$$\frac{I_i^s I_i^e}{I_i^e - I_i^s} > I_{i+1}^e \tag{7}$$

As stated before, the backward solution starts with finding projection origins of I_{n-1} from I_n. According to Property 1, $|\psi_{n-1}|$ will be at most 2. The next step is to find projection origins of I_{n-2} from ψ_{n-1} which contains effective projection origins of I_{n-1} for harmonic period assignment. Considering each member of ψ_{n-1} as a separate interval, it is possible to obtain the resulting ψ_{n-2} by merging projection origins of I_{n-2} with $\psi_{n-1,1}$ and $\psi_{n-1,2}$ into one set as ψ_{n-2}. This step can be repeated until either for one of intervals, ψ_i becomes empty, or we reach ψ_1. The former case means that no harmonic period assignment exists between given intervals.

Alg. 1 presents the existence test for the harmonic period assignment problem. At this step, we assume that there is no intersection between the given intervals, i.e., for all i, $1 \leq i \leq n$, $I_i^e < I_{i+1}^s$. Line 12 of Alg. 1, constructs a temporary set of projection origins of I_i from $\psi_{i+1,j}$. According to Property 1, the resulting projection origin has at most 2 members. However, since according to (2), ψ_i must partition I_i into disjoint sub-intervals, we need to merge the resulting intervals with what already exists in set ψ_i (Line 16 of the algorithm).

In Alg. 1, Line 7 verifies satisfaction of case Fig. 3-(a) where all values in ϕ_i^{a+1} have intersection with $\phi_{i+1,j}$. Also, Line 11 verifies the existence of any integer multiplier between I_i and $\phi_{i+1,j}$. According to [17], if $a^s > a^e$, there is no integer multiplier between two intervals.

Hypothetically, in the worst-case it might be possible that at each step (for each interval I_i), the number of disjoint sub-intervals in ψ_i is $2 \times |\psi_{i+1}|$ because according to Property 1, there are at most 2 projection origins from one interval. Thus, if the merge process in Line 16 does not merge the resulting

Algorithm 1 Existence Test for Harmonic Period Assignment

Input I: where I is the set of intervals
Output {**yes, no**}, ψ: ψ is the resulting projection origins

```
1:  ψ_n ← I_n
2:  ψ ← {ψ_n}
3:  for i ← n − 1 downto 1 do
4:      ψ_i ← ∅
5:      for j ← 1 to h_{i+1} do
6:          calculate a^s and a^e for intervals I_i and ψ_{i+1,j}
7:          if ((a^s + 1)I_i^e ≤ ψ_{i+1,j}^e) then
8:              ψ_i ← I_i
9:              break
10:         end if
11:         if (a^s ≤ a^e) then
12:             construct temporary ψ'_i using a^s and a^e
13:             ψ_i ← ψ_i ∪ ψ'_i
14:         end if
15:     end for
16:     merge intervals in ψ_i so that (2) holds
17:     if (ψ_i = ∅) then
18:         return no
19:     end if
20:     ψ ← ψ ∪ {ψ_i}
21: end for
22: return yes, ψ
```

sub-intervals, their number grows exponentially by the power 2. However, in the next property we show that the number of projection origins of I_i from ψ_{i+1} is at most $|\psi_{i+1}|+1$. In other words, at each step, regardless of the number of temporary projection origins produced by Line 12, the resulting ψ_i will have at most $|\psi_{i+1}| + 1$ members.

Property 2. The number of disjoint sub-intervals inside projection origins of interval I_i from ψ_{i+1} is at most $|\psi_{i+1}| + 1$, i.e., $h_i \leq h_{i+1} + 1$, if the overlapping integer multiples of I_i with I_{i+1} have no intersections.

Proof: We use an induction approach. As the base of induction we show h_{n-1} is at most 2. As shown in Line 1, $\psi_n = I_n$, thus there is only 1 interval in ψ_n. According to Property 1, the number of projection origins between two intervals (here between I_n and I_{n-1}) is at most 2, thus, $h_{n-1} \leq 2$. We assume that for any interval I_j between I_n to I_{i+1}, $1 \leq i < n-1$, $h_j \leq h_{j+1} + 1$.

Now we must prove that $h_i \leq h_{i+1} + 1$. As shown in Fig. 4-(a), if for each member of ψ_{i+1} we have 2 members in temporary set ψ'_i, there will be h_{i+1} *head*s and h_{i+1} *tail*s of I_i in ψ_i. A head is defined as a sub-interval of I_i that contains I_i^s. Similarly, a tail is a sub-interval of I_i that contains I_i^e. Since all of the heads intersect with each other, after the merge operation in Line 16 of Alg. 1, only one head remains which starts from I_i^s and covers the largest value of the ending points of the other heads. This happens for the tails as well, thus, as shown by Fig. 4-(a), the resulting h_i is at most 2.

Because of the fact that if one of $\psi_{i+1,j}$ ($1 \leq j \leq h_{i+1}$) intersects with two disjoint intervals, one of them includes the head and the other includes the tail of I_i, in the worst-case, $\psi_{i+1,j}$ can only contribute 1 more member to ψ_i. This new member in ψ_i, is either a head or a tail. If any other $\psi_{i+1,k}$,

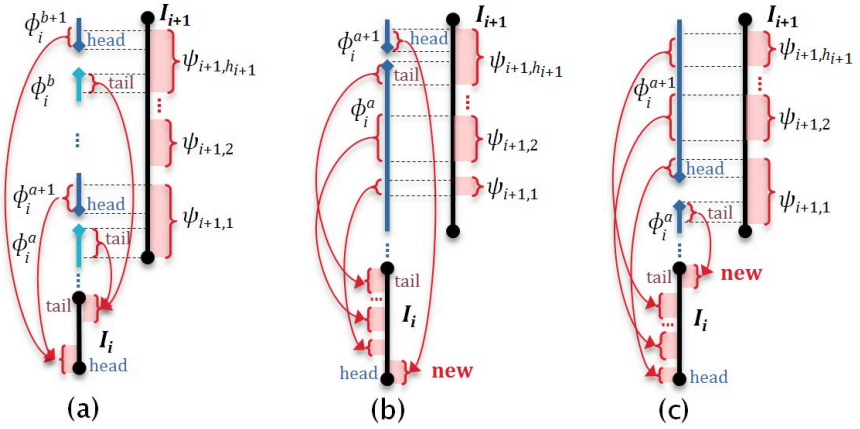

Fig. 4. Examples to show that the maximum number of resulting projection origins of I_i from ψ_{i+1} is at most $|\psi_{i+1}| + 1$

$k \neq j$ intersects with a head or a tail, these heads will be merged with the previous head and do not contribute to h_i. Consequently, in the worst-case, we have one of the scenarios shown in Fig. 4-(b) or (c), and hence, h_i is smaller than or equal to $h_{i+1}+1$. Note that it is possible to have more integer multiples of I_i instead of ϕ_i^a in Fig. 4-(b), however, in the worst-case scenario, their head and tail must not intersect with ψ_{i+1}. Otherwise they will be merged with each other. ■

From Property 2, it is possible to deduce that the maximum number of projection origins for any of the given intervals will not be larger than n because $h_n = 1$, $h_{n-1} \leq 2$, $h_{n-2} \leq 3$, and $h_1 \leq n$. As a result, the for-loop in Lines 5 to 15 is $O(n)$ and the merge process in Line 16 is $O(nlog(n))$. Consequently, because of the for-loop in Lines 3 to 21, computational complexity of the test is $O(n^2log(n))$. Note that to implement the merge process, we sort the intervals in order of increasing start times. Once the intervals are sorted, merging takes linear time because if an interval does not overlap with the start of the next interval, it will not overlap with the other remaining intervals as well.

It is important to note that if the overlapping integer multiples of I_i with I_{i+1} are not distinct, Alg. 1 still works properly as it obtains all projection origins of each of the members of ψ_{i+1}. However, in that case, the number of projection origins will not necessary follow Property 2 and might grow exponentially. An example is shown in Fig. 5. As stated before, if all integer multiples of an interval are overlapping, our previous approach in [17] provides a linear-time solution. In Sect. VI, we elaborate on the cases where the problem can be solved with reasonable computational complexity. In the next step we show that Alg. 1 provides necessary and sufficient conditions of the existence of a harmonic period assignment.

Theorem 1. *Alg. 1 provides necessary and sufficient conditions for the existence of a harmonic period assignment if the given intervals do not intersect.*

Proof: First we show that Alg. 1 provides a sufficient condition. Starting from the last interval, in each step we obtain projection origins of the previous interval. If the for-loop in Lines 3 to 21 is finished successfully, ψ_1 is not empty. Since

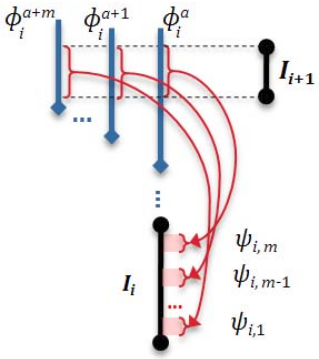

Fig. 5. An example which shows how the number of overlapping integer multiples, denoted by m, may affect the number of projection origins

according to condition (1), for all $x \in \psi_1$, there exists at least one $y \in \psi_2$ such that $y/x \in \mathbb{N}$, there will be at least one possible harmonic assignment between these two intervals. Similarly, based on (1), for the obtained y which belongs to ψ_2, there exists $z \in \psi_3$ such that $z/y \in \mathbb{N}$, and so on. Thus, if ψ is not empty at least one harmonic assignment exists.

To prove that Alg. 1 provides the necessary conditions, we show that at each step, there is no harmonic assignment for intervals I_i to I_n that cannot be found inside ψ_i to ψ_n which are produced by the algorithm. We use contradiction for the proof. For an arbitrary harmonic assignment T_i, T_{i+1}, ..., and T_n where $T_{i+1}/T_i \in \mathbb{N}$, $T_{i+2}/T_{i+1} \in \mathbb{N}$, ..., $T_n/T_{n-1} \in \mathbb{N}$. We assume that all of these values except one of them, i.e., T_m which is selected from $I_m - \psi_m$ ($i \leq m < n$), belong to their respective projection origins which is obtained by the algorithm. According to the assumption, $T_{m+1}/T_m = a \in \mathbb{N}$ and $T_{m+1} \in \psi_{m+1}$. It means that there was one value such as T_m in ϕ_m^a that overlaps with ψ_{m+1} but could not be found by the algorithm. Since intervals do not have intersections, we have $T_1^{min} \leq T_1^{max} \leq T_2^{min} \leq T_2^{max} \leq \ldots \leq T_n^{max}$. Thus, according to Lemma 1 of [17], formula (5) and (6) are able to find all integer multipliers between the intervals. It means that it is not possible that $T_m \notin \psi_m$, or equivalently, it is not possible that Alg. 1 cannot find a. Since according to Line 1

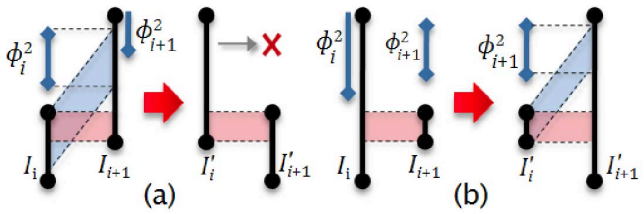

Fig. 6. Two basic cases of overlapping intervals

of the algorithm, $\psi_n = I_n$, at each step of the algorithm, all possible values which can be origins of a harmonic assignment will be found. Thus, a harmonic assignment which does not belong to ψ_m does not exist. ■

In the next subsection we show that Theorem 1 holds even for overlapping intervals.

B. Handling Overlapping Intervals

In this sub-section, we show that our solution approach is not influenced by the task indexing assumption, i.e., $T_1^{min} \leq T_2^{min} \leq \ldots \leq T_n^{min}$. However, hypothetically, the indexing method can remove some of the feasible harmonic assignments which could be found if another ordering would have been applied. Consequently, the intervals order is important if they overlap with some other intervals. To show that it is not required to consider different task orderings in our solution approach, we discuss all overlapping cases between two intervals I_i and I_{i+1} shown in Fig. 6.

Lemma 1. *Re-indexing two intervals I_i and I_{i+1} with $I_i^s < I_{i+1}^s$ and $I_i^e < I_{i+1}^e$ will not add any potential harmonic assignment to the final solution in ψ obtained by Alg. 1.*

Proof: We denote the new indexing by $I'_i = I_{i+1}$ and $I'_{i+1} = I_i$. According to formula (5) and (6), $a^s = 1$ and $a^e = \lfloor I_{i+1}^e / I_i^s \rfloor$ while if the order is reversed, $a'^s = 1$ and $a'^e = \lfloor I_i^e / I_{i+1}^s \rfloor$. Hence, the resulting set of integer multipliers in the new ordering will be a subset of the previous case because $a'^e < a^e$. The reason is that no integer multiplier exists which can map a value in $(I_i^e, I_{i+1}^e]$ to a value in $[I_i^s, I_i^e]$. Consequently, by re-indexing the intervals, no new harmonic assignment can be added to ψ. ■

Unlike the first case, if we re-order two intervals I_i and I_{i+1} in Fig. 6-(b), new projected harmonic zones may appear which could not be found before. Hypothetically, these new projected zones can be a source of new harmonic relations which were ignored by the previous indexing. However, by the following lemma and theorem we show that even without considering other possible orderings for those overlapping intervals, Alg 1 still provides necessary and sufficient conditions for the existence of a harmonic period assignment.

Lemma 2. *Projection origins of I_i from I_{i+1}, $I_i^s \leq I_{i+1}^s$ are a super set of projection origins of $I'_i = I_{i+1}$ from $I'_{i+1} = I_i$.*

Proof: We show that ψ'_i will be a subset of ψ_i. In the new ordering, $\psi'_i = I'_i \cap \psi'_{i+1}$ because the first integer multiple of I'_i which overlaps with I'_{i+1} is ϕ_i^1 and it is totally covered by I'_{i+1}. On the other hand, if the previous indexing is used, we have $\psi_{i+1} \subset I_{i+1}$, thus, the effective projection origins of I_i from I_{i+1} will appear in the sub-interval $[I_i^s, I_{i+1}^e]$. Consequently, $\psi_i = \{I_i \cap \psi_{i+1}\} \cup A$ where A is a set of projection origins of $[I_i^s, I_{i+1}^s)$ from ψ_{i+1}. Assuming $\psi'_{i+1} = I'_{i+1}$ as the largest possible ψ'_{i+1}, we have $I'_i \cap \psi'_{i+1} = I'_i$ which means that $\psi_{i+1} = I_{i+1}$, and hence, $I_i \cap \psi_{i+1} = I'_i \cap \psi'_{i+1}$. However, due to the fact that $\psi_i = I_i \cap \psi_{i+1} \cup A$, it will be a super set for ψ'_i which is equal to I'_{i+1}. ■

Now we show that Theorem 1 holds even if the intervals are overlapping.

Theorem 2. *Alg. 1 provides necessary and sufficient conditions of the existence of a harmonic period assignment between given set of intervals.*

Proof: The proof of sufficiency is the same as Theorem 1 because if Alg. 1 returns ψ with non-empty ψ_i members, at least one harmonic assignment can be found inside the resulting projection origins. Also Lemma 1 and 2 show that the size of ψ_i will not be increased if another indexing is used between any two overlapping intervals I_i and I_{i+1}. As a result, we can use the same proof as Theorem 1 to show necessity of Alg. 1. ■

It is important to note that if another indexing method is used for the overlapping intervals such as Fig. 6-(b), the resulting ψ from Alg. 1 will only contain *a subset of* all possible period assignments. It means that there might be some assignments which cannot be found by our indexing method. These neglected possible assignments may contain solutions with lower utilization because as shown by Fig. 6-(b), after re-indexing the tasks we might be able to assign larger period to I'_{i+1} than the previous case.

C. Handling Integer Value Periods

In most of real-time systems, task periods have to be integer values, which describe multiples of the system's clock. However, if during the construction of projection origins, it is allowed to have non-integer starting and ending values for each $\psi_{i,j}$, non-integer periods may be selected in the final period assignment. As shown in Fig. 7, to prevent this situation, it is enough to force $\psi_{i,j}^s \leftarrow \lceil \psi_{i,j}^s \rceil$ and $\psi_{i,j}^e \leftarrow \lfloor \psi_{i,j}^e \rfloor$.

The result of dividing a non-integer value by an integer divisor will not yield an integer. Similarly, if an interval such as $[10.001, 10.999]$ is divided by any arbitrary integer value, the resulting interval will not contain any integer numbers. Thus, if the intersection between an integer multiple of I_i does not have any integer values (such as ϕ_i^a in Fig. 7), the respective projection origin will not have any integer numbers, and consequently, will not be added to ψ_i since it becomes an empty set. Also, to have an integer period assignment, it is not necessary to keep $\psi_{i,j}$ as a set of discrete integer numbers because if during the creation of ψ, one of the intervals intersects with the non-integer parts of $\psi_{i,j}$, it will be automatically removed from its projection origins.

Since for two integer numbers $x, y \in \mathbb{N}$, we have $xy \in \mathbb{N}$, by rounding projection origins of each interval at each step of Alg. 1, we can assign harmonic integer periods only if the first period is selected from the integer values in ψ_1. Due to condition (1), for all values including $T_1 \in \mathbb{N}$ in ψ_1, there

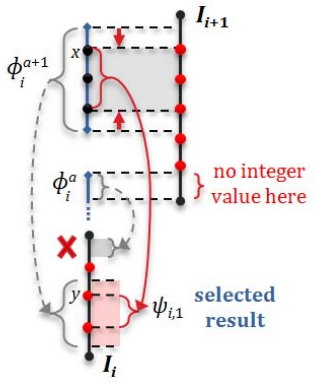

Fig. 7. An example to show how to select ψ_i which is bounded to integer values

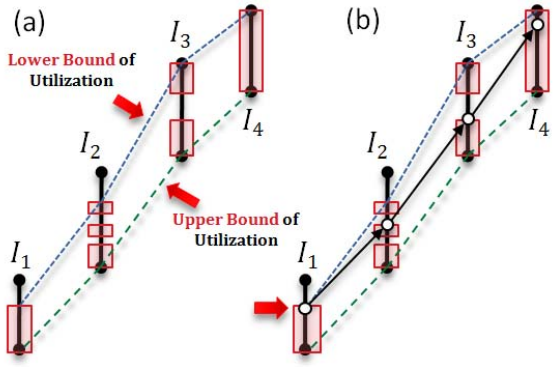

Fig. 8. An example for ψ; a) utilization bounds of potential solutions, b) a possible period assignment.

exists a value T_2 in ψ_2 such that $T_2/T_1 \in \mathbb{N}$, thus, $T_2 = aT_1 \in \mathbb{N}$ where a is an integer multiplier.

If the projection origins are not reduced to integer numbers, a harmonic period assignment with integer numbers might not exist because the intersection between two intervals may only consist of non-integer values, and hence, it may not be possible to assign an integer value from the respective projection origins. To guarantee such an assignment, the rounding process must be done during the construction of the super-set ψ.

V. Harmonic Period Assignment

In this section, we first derive lower and upper bounds of the utilization of potential period assignments where none of the given intervals overlap with each other. Aiming to construct low utilization task sets, we present a heuristic period assignment algorithm and sufficient conditions to provide $U \leq 1$. Here, we call this assignment *utilization-feasible harmonic period assignment* (UF-HPA).

A. Utilization Bounds of the Feasible Solution

We apply the idea of our previous work [17], to derive utilization bounds of all potential harmonic period assignments for the task sets with no overlapping period ranges. As shown in Fig. 8-(a), for the obtained set ψ, we have

$$U^{lb} \leq U \leq U^{ub} \tag{8}$$

where U^{lb} and U^{ub} are lower and upper bounds of the utilization as

$$U^{lb} = \sum_{i=1}^{n} \frac{c_i}{\psi^e_{i,h_i}} \tag{9}$$

$$U^{ub} = \sum_{i=1}^{n} \frac{c_i}{\psi^s_{i,1}} \tag{10}$$

It is important to note that a harmonic period assignment that guarantees U^{lb} (or U^{ub}) might not exist because it requires $\psi^e_{i+1,h_{i+1}}/\psi^e_{i,h_i} \in \mathbb{N}$ for all $1 \leq i < n$.

If the intervals are overlapping similar to what is shown in Fig. 6-(b), re-indexing the tasks affects the lower bound of the utilization because it might be possible to assign a larger period to τ_i from I_i. In that case, formula (9) becomes a conservative lower bound of the utilization, and the actual lower bound might be smaller. Note that re-indexing will not affect our theoretical upper bound because if our indexing method is applied on Fig. 6-(b), it forces I_i to have a relatively high period value which is smaller than or equal to the period of I_{i+1}. However, according to Lemma 1, in cases such as Fig. 6-(a), by re-indexing the intervals, no new assignment will be added, and hence, it has no effect on the utilization bounds as well.

B. A Heuristic Algorithm for Period Assignment

The idea of our heuristic algorithm is to assign the largest harmonic period that can be obtained from the previous period for each task. This algorithm is presented in Alg. 2. In the first step, it assigns $T_1 \leftarrow \psi_{1,h_1}$. Then, for each task τ_i, it finds the largest harmonic period in ψ_i such that T_i/T_{i-1} is harmonic with multiplier a^e which is the largest multiplier that maps T_{i-1} to ψ_i. Since according to (1), for every x in ψ_{i-1} there exists at least one value y in ψ_i such that $y/x \in \mathbb{N}$, Line 5 of the algorithm will be true for at least one of ψ_i members. To find a low utilization assignment, we search from ψ_{i,h_i} down to $\psi_{i,1}$, thus, the first T_i which is assigned in Line 6, will be the largest one which is reachable from T_{i-1}. Note that the algorithm must start from ψ_1, otherwise, the existence of a harmonic assignment cannot be guaranteed.

The computational complexity of Alg. 2 is $O(nH)$ where $H = \max\{h_1, h_2, \ldots, h_n\}$. If conditions of Property 2 hold, H will be bounded to n, and as a result, Alg. 2 runs in $O(n^2)$.

Note that as shown in Fig. 8-(b), Alg. 2 might not assign ψ^e_{i,h_i} to T_i. Consequently, it cannot guarantee $U \leq 1$ as the periods are not the largest to assign. However, if the following condition holds, Alg. 2 provides a utilization-feasible assignment.

Theorem 3. *If for the given set ψ from Alg. 1, we have $U^{ub} \leq 1$ where U^{ub} is obtained from (10), Alg. 2 always selects utilization-feasible harmonic periods.*

Proof: The proof is trivial because if $U^{ub} \leq 1$, even if Alg. 2 assigns the smallest periods to the tasks, the utilization restriction is still satisfied. ■

Algorithm 2 Heuristic Algorithm for Harmonic Period Assignment

Input ψ
Output $\{T_1, T_2, \ldots, T_n\}$

$T_1 \leftarrow \psi^e_{1,h_1}$
for $i \leftarrow 2$ **to** n **do**
  for $j \leftarrow h_i$ **downto** 1 **do**
    $a^e \leftarrow \lfloor \psi^e_{i,j} / T_{i-1} \rfloor$
    if $(a^e T_{i-1} \geq \psi^s_{i,j})$ **then**
      $T_i \leftarrow a^e T_{i-1}$
      break
    end if
  end for
end for
return $\{T_1, T_2, \ldots, T_n\}$

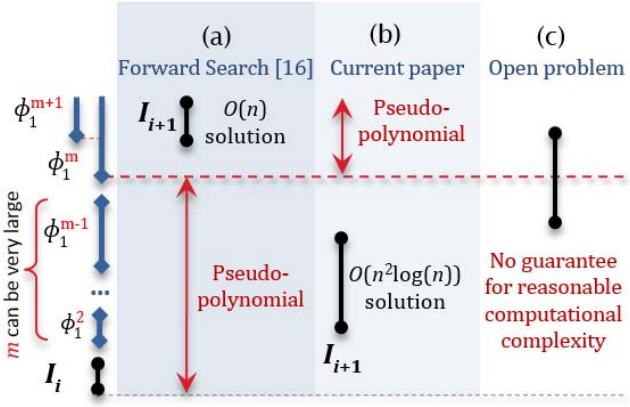

Fig. 9. Summary of the computational complexity of: a) our previous work [17], b) this paper, and c) open problem. In this figure, m is the number of non-overlapping integer multiples.

VI. Discussions

In this section, first we summarize our achievements and open problems and then discuss UF-HPA. Fig. 9 summarizes the computational complexity of our period assignment algorithms in this paper (backward solution) and the previous one (forward solution) [17]. As shown in the figure, if $A < I^s_{i+1}$ where $A = (I^s_i I^e_i)/(I^e_i - I^s_i)$, the solution in [17] runs in $O(n)$ while the backward solution may have considerable computational complexity (this condition has been shown by a dashed horizontal line in the middle of the figure). However, in these cases, any value in I_{i+1} will be in a harmonic relation with one of the values of I_i, which inherently reduces the complexity of the problem itself. In this paper, we have focused on the cases where $A > I^e_{i+1}$ because in those cases, only a subset of sub-intervals of I_{i+1} will be in a harmonic relation with a subset of sub-intervals of I_i. As we have shown, our new solution is $O(n^2 log(n))$ and even if the given intervals intersect with each other, our existence test is still valid. Lastly, the computational complexity of the cases with $I^s_{i+1} \leq A \leq I^e_{i+1}$ and the actual hardness of the harmonic period assignment problem have remained as open problems.

Next we discuss why it is not easy to find an optimal UF-HPA. As shown by Fig. 10, the utilization of τ_2 is

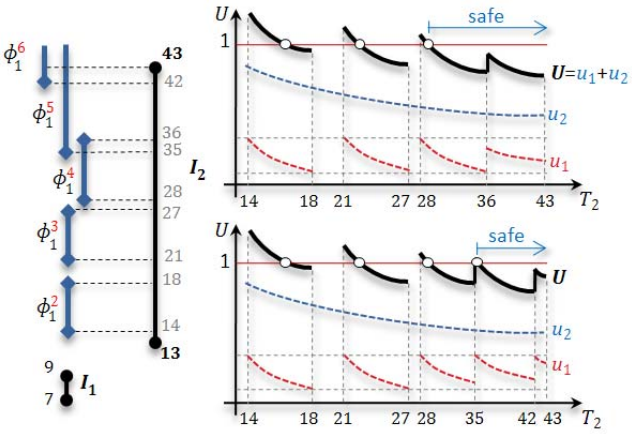

Fig. 10. Utilization of two tasks $\tau_1 : (2, [7, 9])$ and $\tau_2 : (10, [13, 43])$ as a function of T_2 based on two possible choices for T_1; a) larger T_1, b) smaller T_1.

not necessarily a strictly decreasing function of T_2. Besides, a utilization-feasible assignment exists only for some sub-intervals of I_2 where $u_1 + u_2 \leq 1$. Because of the fact that the utilization is a decreasing function of T_1 (or T_2), to find the starting point of a feasible solution in each sub-interval, we must calculate the shortest T_1 and T_2 values that construct a feasible utilization. The white circles in the figure present those starting points. In the intersection between ϕ^a_1 with I_2, the first feasible T_1 and T_2 can be calculated by the following formulas

$$\max\{aI^s_1, I^s_2, ac_1 + c_2\} \leq T^f_2 \leq \min\{aI^e_1, I^e_2\} \quad (11)$$

$$T^f_1 = T^f_2 / a \quad (12)$$

Note that in the upper curve of Fig. 10, if utilization of one of the splitting points in the curve, such as $x = 36$, falls under 1, any other $T_2 \geq x$ will have a feasible utilization because the utilization function is decreasing with the increase in T_1 and T_2. Note that the upper curve is obtained from selecting larger values of T_1 whenever there were more options for T_1. Another observation is that the smallest utilization is either the last or the one before the last falling point in the utilization curve, i.e., it is either at 36 or at 43 in our example. The reason is that u_2 is decreasing with the increase in T_2. Consequently, the minimum U appears in either of these two cases: a) at 36 where we have minimum u_1 and relatively small value of u_2 or b) at 43 where we have minimum u_2 and relatively small u_1. If the minimum utilization is still greater than 1, a UF-HPS does not exist for these two tasks.

However, because each projection of an integer multiple of I_1 to I_2 has different utilization range, in an optimal solution we might need to consider all of those integer multiples. As it has been shown in our previous work [17], the number of distinct integer multiples between two intervals can be exponential w.r.t the number of intervals. Thus, a search-based solution might have considerably high computational complexity. As the problem of harmonic period assignment is not linear, i.e., it can be mapped to an integer non-linear optimization problem, finding an optimal UF-HPA solution

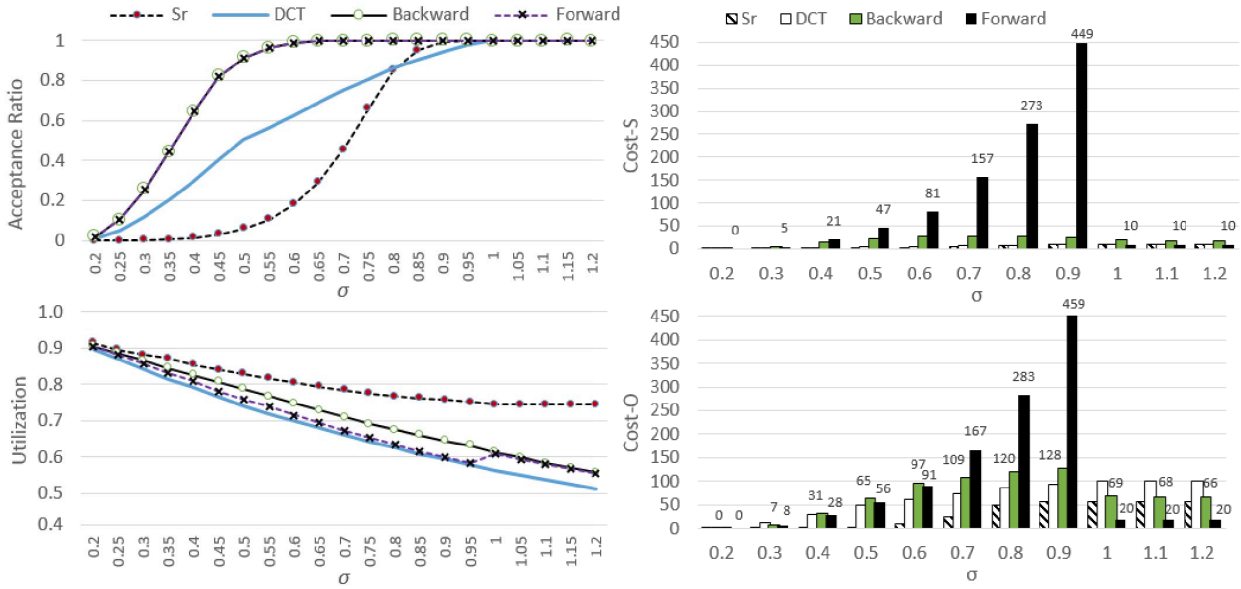

Fig. 11. Acceptance ratio, utilization, Cost-S, and Cost-O of the algorithms for different values of σ. Acceptance ratio of the forward and backward approaches is overlapped.

with reasonable computational complexity might not be easy, and the actual hardness of the problem is still in question.

VII. Experimental Results

In this section we evaluate performance of our backward approach (according to Alg. 2) against the forward approach [17] as well as Sr and DCT algorithms [4]. The goal is to produce feasible harmonic task sets with low utilization. Performance measures are acceptance ratio, which is the ratio of harmonic task sets with $U \leq 1$ to the total number of generated task sets, and average utilization of successfully generated harmonic task sets. Besides, we measure data structure size (denoted by Cost-S), and the number of operations (denoted by Cost-O) which is required to produce the final period assignment. In the latter measure, the focus is to count the operations affecting computational complexity of the algorithms, thus, Lines 6 and 7 of Alg. 1 are not counted while Line 8 is one operation and Line 12 is $1 + a^e - a^s$ operations. With the same intuition, in the merge procedure (Line 16) we may have at most $|\psi_i|^2$ operations where $|\psi_i|$ is the size of ψ_i in its intermediate stage, i.e., before it becomes a set of disjoint intervals. On the other hand, by Cost-S we measure final data structure size, thus, Cost-S is incremented by $|\psi_i|$ after the merge operations. In other words, we do not consider memory requirements of the intermediate stages. The simulations have been done in the discrete event simulator (DEVS-Suite) [21]. Each experiment is repeated 10000 times, and final outputs are within a confidence interval ± 0.05 for confidence level 0.95.

We have used a modified version of Sr as introduced in [17] because its original version cannot cope with period ranges. 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 $(0.5T_1, T_1]$, however, since in our experiments, T_i is not known in advance, we assign the smallest value with pattern $r2^x$ which is inside $[T_i^{min}, T_i^{max}]$ to T_i, where x is an integer number smaller than $\lfloor log(max\{T_i^{max}\}/r) \rfloor$. Since the choice for r depends on T_1 which is unknown in advance, first we assume $T_1 = T_1^{min}$, and then we gradually increase it to T_1^{max} (by step 0.1 of the previous value) until we find a feasible harmonic period assignment with n tasks.

In the first experiment we consider $n = 10$ tasks, and in the second experiment, we study the effect of tasks n for different values of n. In the first experiment, to generate period ranges, a random T_1^{min} is selected from $[5, 50]$, and then $T_1^{max} = (1+\sigma)T_1^{min}$ where σ shows the wideness of the period ranges with respect to their starting value. Then we generate $T_i^{min} = kT_{i-1}^{min}$ and $T_i^{max} = (1+\sigma)T_i^{min}$ where $k \in \mathbb{R}$ is a random value in $[1.0, 5.0]$ and shows how far are the starting points of two period ranges from each other. Execution times of the tasks are assigned as $c_i = u_i T_i^{min}$, where u_i is generated by the uUniFast algorithm [22] for utilization 1. Thus, the final task set utilization will be 1 only if the period of each task is exactly T_i^{min}.

As shown by Fig. 11, the acceptance ratio of our forward and backward approaches, which are both optimal to find harmonic period assignments, is considerably higher than Sr and DCT. Acceptance ratios of DCT and Sr increase by the increase in the wideness of the period ranges, however, in average, DCT has higher acceptance ratio than Sr because it does not limit itself to the period ratios which are powers of 2. Nonetheless, DCT cannot find harmonic assignments if there is no feasible assignment which include any of T_i^{max} values. That is why, for example in $\sigma = 0.5$, acceptance ratio of DCT is around 50% while in our methods it is 91%. In this experiment, in average, acceptance ratios of Sr, DCT, and our optimal solutions are 50%, 64%, and 80%, respectively.

Although both of the forward and backward approaches use heuristic period assignment algorithms, the final utilization

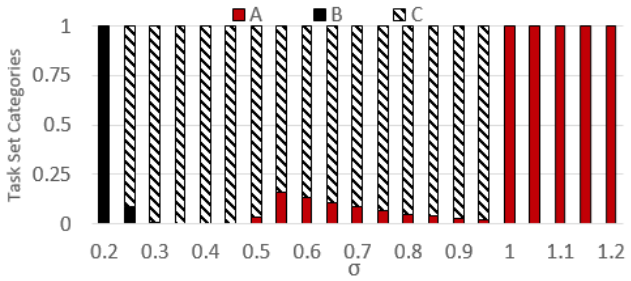

Fig. 12. Distribution of the task set categories based on Fig. 9 in the first experiment

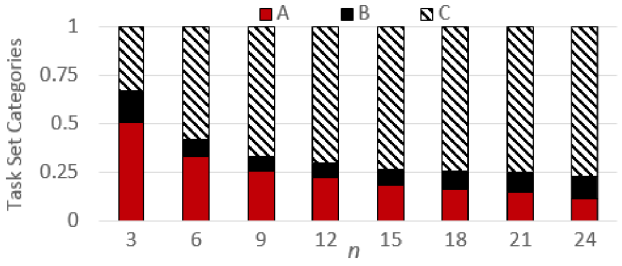

Fig. 14. Distribution of the task set categories based on Fig. 9 in the second experiment

of the feasible task sets in the forward approach is lower than the backward approach because during the construction of the graph, it chooses the path with the lowest potential utilization. Later, this path is used with a heuristic period assignment function while in the backward approach, we do not differentiate between paths. Consequently, the final utilization of the forward approach is lower than Alg. 2 in this experiment, though none of them provide an assignment with the minimum utilization.

As shown by Fig. 11, by the increase in the wideness of period ranges, the chance that the starting value of the ranges satisfies condition (7) increases. Consequently, in the forward approach, size of the graph reduces which leads to significant reduction in both Cost-S and Cost-O for $\sigma \geq 1$. In Fig. 12 we categorize the generated task sets into categories A, B, and C, based on the conditions defined in Sect. VI for Fig. 9. Category A shows the task sets satisfying tight harmonic relations according to [17], category B represents those which satisfy (7), and category C includes the task sets which are neither in A nor in B. As we can see in Fig. 12, most of the generated task sets for $0.3 < \sigma < 1$ fall in category C. Even though we could not theoretically guarantee a polynomial-time computational complexity for backward approach in category C, we see that Cost-S and Cost-O of the backward approach are not exponentially growing, as it happened for the forward approach.

When σ is small, there is a small amount of potentially feasible harmonic assignments due to the limited wideness of the period ranges. It leads to smaller cost for both of our approaches, however, when increasing the wideness of the ranges, the number of disjoint possible feasible solutions increases. That is why the size of the graph in the forward approach increases rapidly. Nonetheless, for high values of σ, many period ranges are so wide that they overlap with the integer multiples of other intervals, and hence, the size of the graph in the forward approach is reduced. In the backward approach, both Cost-S and Cost-O decrease with the increase of σ because the tasks with a larger starting point have larger period ranges than other tasks. As a result, it may happens that they cover *head* or *tail* of integer multiples of other intervals. Consequently, the number of disjoint projection origins reduces and they become larger so that they cover the whole period range of the next task. It leads to smaller Cost-S and Cost-O for our backward approach.

The the parameter of our next experiment is the number of tasks in the task set. We generate the task set as explained earlier, however, this time, σ is selected randomly with uniform distribution from $[0.2, 1.2]$. Fig. 13 shows the results of this experiment. With the increase in the number of tasks, the chance that a harmonic period assignment exists between all of the period ranges decreases. The same as Fig. 11, we see that the forward approach manages to find task sets with lower utilization than the backward approach. This diagram also shows that in our experiments, the utilization of the accepted task sets is almost not affected by n. Even though according to Fig. 14, most of the generated task sets fall in the category A and C, still our backward approach has a reasonable computational- and memory-complexity. When the task set is small, it has higher chance to fall in category A or B. That is why when $n < 12$, forward approach outperforms backward approach in terms of Cost-S and Cost-O. However, with the increase in the number of tasks, computational complexity of the forward approach increases rapidly, which may make it unsuitable for the task sets with large number of tasks.

VIII. Summary and Conclusion

In this paper we have addressed the problem of assigning harmonic periods out of acceptable period ranges for tasks. We use a backward search (from the last period range to the first one) to derive and store effective sub-intervals of the given ranges which can be origins of harmonic relations. Using this approach, the required data structure size reduces efficiently in comparison with our previous algorithm. While in many cases, the previous algorithm exhibits exponential growth during the search process, we have shown here that the problem can be solved in $O(n^2 log(n))$ where n is the number of tasks, if every period range only intersects with non-overlapping integer multiples of the previous range. We have presented an algorithm to verify necessary and sufficient conditions for the existence of a harmonic period assignment. We have shown that this test is not affected if the given period ranges are overlapping. Moreover, we have provided utilization bounds for the potential assignments and introduced a heuristic algorithm to construct low utilization harmonic task sets. We have discussed the existing obstacles to find a utilization-feasible harmonic period assignment as well as the cases where our methods cannot provide affordable solutions.

Acknowledgment

The authors would like to thank Morteza Mohaqeqi for his helpful comments. We are grateful for the comments of the anonymous reviewers of the paper.

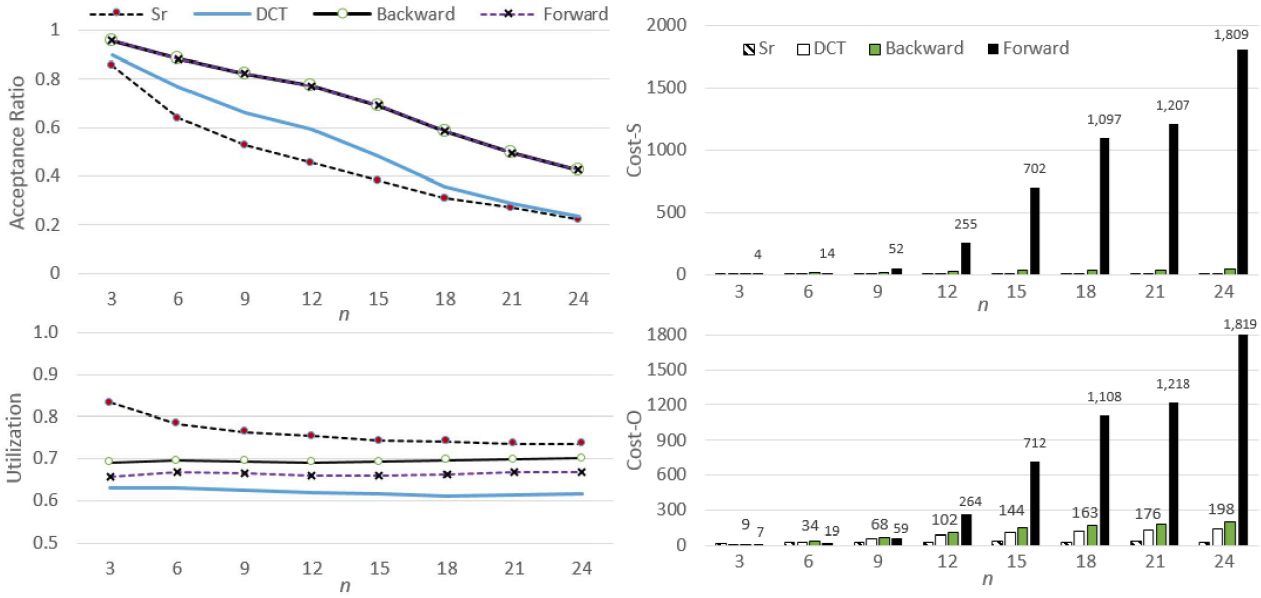

Fig. 13. Acceptance ratio, utilization, Cost-S, and Cost-O of the algorithms for different values of n. Acceptance ratio of the forward and backward approaches is overlapped.

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

[3] M. Joseph and P. Pandya, "Optimal Checkpoint Placement on Real-Time Tasks with Harmonic Periods," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[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] 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.

[6] M. Nasri and M. Kargahi, "Precautious-RM: a predictable non-preemptive scheduling algorithm for harmonic tasks," *Real-Time Systems*, vol. 50, no. 4, pp. 548–584, 2014.

[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] T. Taira, N. Kamata, and N. Yamasaki, "Design and implementation of reconfigurable modular humanoid robot architecture," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 3566–3571.

[11] I. Mizuuchi, Y. Nakanishi, Y. Sodeyama, Y. Namiki, T. Nishino, N. Muramatsu, J. Urata, K. Hongo, T. Yoshikai, and M. Inaba, "An advanced musculoskeletal humanoid kojiro," in *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, 2007, pp. 294–299.

[12] 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.

[13] 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.

[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] S. Kwak and J.-M. Yang, "Optimal Checkpoint Placement on Real-Time Tasks with Harmonic Periods," *Journal of Computer Science and Technology*, vol. 27, no. 1, pp. 105–112, 2012.

[17] M. Nasri, G. Fohler, and M. Kargahi, "A Framework to Construct Customized Harmonic Periods for Real-Time Systems," in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2014, pp. 211–220.

[18] 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.

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

[21] "Discrete-event system simulator (DEVS-suite)," 2014. [Online]. Available: http://acims.asu.edu/software/devs-suite

[22] E. Bini and G. Buttazzo, "Measuring the Performance of Schedulability Tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.