On Flexible and Robust Parameter Assignment for Periodic Real-Time Components

Mitra Nasri
Max Planck Institute for Software Systems (MPI-SWS)
mitra@mpi-sws.org

ABSTRACT

In order to increase the flexibility of design process, we consider a component-based system in which some components may be changed, repaired, or upgraded. We assume that the worst-case execution time (WCET) of each task that is implemented by a component is known. Our first goal is to find the smallest set of periods that make the task set schedulable by EDF or by RM. We call these periods *safe periods* because then any period assignment which is larger than the safe periods will be schedulable. Having safe periods, system designer will be able to use any optimization criteria of his choice to assign periods of the tasks without a need to incorporate the response time analysis techniques, which are usually NP-Hard problems, into his optimization constraints. Instead, the schedulability constraint will be reduced to verifying that the assigned periods are larger than the safe periods. Our solution for safe periods for the RM, is based on finding the smallest set of harmonic periods with utilization 1. Here we use our recently developed polynomial-time approximation algorithm with bounded error of 2 for finding these safe periods for RM.

As the second part of our contribution, we consider the robustness property. First we explain how to find safe periods such that a certain level of robustness (based on potential changes of each component) is guaranteed. Doing this, if one of the components changes in the future, other components can still run using their previous periods, and hence, we isolate the effect of future changes from parameters of the other components. Finally, we obtain the robustness factor as a function of the available spare capacity of the system, i.e., the unused utilization. We determine to what extend the WCET of any of the components can be increased without violating the safe periods.

Keywords: *Robustness, Schedulability Guarantee, Parameter Assignment, EDF, RM*

1. INTRODUCTION

With the increase in the computational power of underlying platforms, real-time systems tend to have more functionalities than before. In order to reduce time and costs of developing such large scale systems, component-based design has been widely adopted by the industries. It allows a new system to reuse what has been developed in the previous projects or by other companies or third parties. In the case of safety-critical system, reuse of previously certified components simplifies the re-certification process as well. Eventually, it saves time and money in the design of a new system.

During the design of a large scale system, component's (or task's) characteristics such as execution time, period, or deadline may change. Even in some cases, new requirements or tasks will be added to or removed from the system. As a result, schedulability analysis method, which is the theoretical means to guarantee real-time requirements, must be flexible towards changes of its input parameters. Moreover, due to the fact that the system evolves, accurate information about the parameters of the tasks might not be available during all design stages. In some cases, a task might have a set of configurations that are not yet confirmed (finalized). The schedulability analysis method must also cope with these situations as well.

In a system that consists of several components, assigning the parameters of the components (or tasks) is a challenging step; some components may have certain restrictions for their period or deadline. If there are a large number of components and the initial parameter assignment does not pass the schedulability test, then the designer must change the parameters until he finds a feasible configuration for the system. Moreover, some set of parameters may not satisfy other design goals. For example, a schedulable set of parameters may heavily under-utilize the system, or decreases quality of service.

One potential solution for finding a configuration which satisfies all design goals is to use *design space exploration* [12] which provides the ability to operate on the space of design configurations in order to evaluate and/or find feasible configuration sets (or regions). This exploration needs a mathematical model of the system and its constraints. They are usually integrated into one constrained optimization problem and are solved by numerical solvers. However, a large system may have millions of possibilities, and hence, it might not be feasible to enumerate every point in the design space. Besides, it is hard (and probably expensive) to derive one big mathematical model of a large system which can be fed to the tool set. Another issue is that adding or removing components may force the system to under-go another round of design space exploration during which timing parameters of some other tasks may change again.

Another important challenge in the design of component-based systems is that components may be upgraded. Due to these upgrades, their worst-case execution time (WCET) may also change. Such a change may require redoing the schedulability analysis of the system (in most cases), because most of the schedulability tests are not *robust* against an increase in the work-demand of a task [2]. It is worth noting that traditional response-time analysis for rate monotonic (RM) and demand bound analysis for EDF are *sustainable* towards reductions in the WCET (or an increase in the pe-

riod) either at run-time or design-time. In other words, if the new component has smaller WCET, system remains schedulable [2]. However, they are not robust against the increase in the WCET.

One way to provide robustness is through *sensitivity analysis* which determines to what extent any of the parameters (or a combination of parameters) can be increased while the schedulability is preserved [5]. However, the works that has provided a means for the sensitivity analysis [3, 5] are computationally expensive because they are based on building and then solving a large set of equations from the current system parameters. If the system has a large number of tasks, the problem becomes intractable.

In this paper, we assume that the system consists of a set of components that implement one or more real-time tasks. Each task is introduced by its initial WCET. Also we assume that these tasks are periodic or sporadic, have implicit deadlines, and are scheduled upon one processor either using RM or EDF scheduling algorithm which are the two widely implemented scheduling algorithms in real-time operating systems. The goal of our paper is to provide a method for finding a *safe* set of periods such that any final period T_i that is assigned to the task τ_i and is larger than the safe period T_i^S, does not threaten the schedulability of the whole system. In other words, periods can be selected independently from each other as long as each period is larger than T_i^S. This property provides freedom for the system designer to decouple the schedulability analysis from optimization criteria. Thus, it is a flexible solution because then the designer can use optimization methodologies of his/her choice without a need to add separate conditions for the schedulability guarantee within the optimization criteria (apart from having $T_i \geq T_i^S$). Since the response time analysis problem is known to be NP-Complete [9, 10], this simplification can significantly help to reduce overall cost of optimization process.

As our second contribution, we provide robustness guarantees for the system which is subject to changes (in the future) provided that either the bounded amount of changes per task, or overall amount of changes per all tasks is given. In other words, we require that the designer gives us a list of α_i values per task τ_i that is the maximum potential change that can happen for the WCET of that particular task. This can happen because of a change in the implementation or in the input parameter of the task. For example, an application uses a camera to obtain video in VGA format with image size 640x480, but it may happen that the next version of the application uses image size 1024x768 which increases the execution time of the video processing component by 10% (for instance). Meanwhile, the system designer wants to be sure that the provided set of T_i^S values by our solution are still valid even if such changes happen. The second contribution makes it possible to provide T_i^S values with such guarantees for the future changes. However, unlike the first step, here our solution may under-utilize the system due to the fact that it needs to leave some gap for the potential future changes.

As our third contribution, we answer to the question that "how much robustness can be guaranteed if the system has a specified spare utilization?". Here, for a task τ_i we provide α_i such that if the amount of changes in C_i (WCET of τ_i) is smaller than $\alpha_i C_i$, the provided T_j^S values remain valid for all tasks. Another application of this contribution is to help designers to know the potential effects of removing a component from the system.

As the final contribution, we explain how to add more tasks to the system while keeping all previous promises about safe periods, i.e., T_i^S values. In this paper, we try to keep the complexity of the solution polynomial-time. In the case of safe period assignment for EDF, our complexity is linear time and in the case of the RM is $O(n^2)$, where n is the number of tasks.

The paper is organized as follows. We start by introducing the system model and notations in Sect. 2 and introduce our definitions and concepts in Sect. 3. In Sect. 4 our solution is introduced and is evaluated in Sect. 5. In Sect. 6 related work is discussed, and the paper is concluded in Sect. 7.

2. SYSTEM MODEL

We assume a uni-processor system with a set of independent hard real-time preemptive tasks denoted by $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ which are provided by a set of components that may or may not be open source. However, we assume that the initial WCET of each task in each component is given, and is denoted by C_i for the task τ_i. Also we assume that the tasks are either periodic with no release offset and jitter, or they are sporadic. Moreover, we consider that the tasks have implicit deadlines, i.e., deadlines are equal to periods. We denote the final period of a task by T_i. Also we denote the utilization of a task by $u_i = C_i/T_i$ and utilization of the system by $U = \sum_{i=1}^{n} u_i$. Finally, we assume tasks are sorted in a non-decreasing order of their WCET, i.e., $C_1 \leq C_2 \leq \ldots \leq C_n$.

The system will be scheduled by either RM or EDF scheduling algorithms and the preemption overheads is negligible. We assume that tasks do not share resources and do not have self-suspension (these cases are left for the future work).

3. DEFINITIONS AND CONCEPTS

The first goal of our work is to identify a set of *safe* period values $\{T_1^S, \ldots, T_n^S\}$ that simplify the schedulability condition to a bound-checking condition, i.e., instead of a schedulability test, we just need to make sure that $\forall i; T_i \geq T_i^S$ in order to guarantee the schedulability. Safe periods are defined below. Note that here we did not mention the scheduling algorithm in the definitions, but later we will study EDF and RM separately.

DEFINITION 1. *For a given set of WCET, a set of periods $\{T_1^S, \ldots, T_n^S\}$ is a safe period set if and only if every task in the task set is schedulable as long as*

$$\forall i;\ T_i \geq T_i^S. \tag{1}$$

If the periods are assigned such that (1) is satisfied, then the maximum possible utilization that can be produced by the task set will be not larger than $\sum_{i=1}^{n} C_i/T_i^S$. A given set of safe periods is called U^S-safe if $U^S = \sum_{i=1}^{n} C_i/T_i^S$.

Among different choices for T_i^S values, we are interested in the smallest ones because then we maximize the number of possible period values that can be used by the designer. Based on this intuition, we define our optimality criteria as follows.

DEFINITION 2. *A given set of $\{T_1^S, \ldots, T_n^S\}$ is called optimal U^S-safe period if and only if it is U^S-safe and it*

minimizes the following value

$$\sum_{i=1}^{n} w_i T_i^S \tag{2}$$

where w_i shows the importance of having T_i as close as possible to C_i and $0 < w_i \leq 1$.

If we have $\forall i; w_i = 1$, equation (2) will not prioritize one task over another. Namely, the effect of weights is gone and all tasks are treated the same way. However, if for example, it is very important to have T_1 as close as possible to C_1, but for the rest of the task it is not important, then we can assign $w_1 = 1$ and for the other tasks $w_i \approx 0$.

In order to formalize the robustness guarantee, we define a bounded guarantee for the robustness. We start with α_i-robust periods and then $\alpha-$robust periods.

DEFINITION 3. *Given a set $A = \{\alpha_1, \ldots, \alpha_n\}$, a period assignment T^S is called α_i-robust if and only if $\forall i; 1 \leq i \leq n$; it is possible to increase C_i upto $\alpha_i C_i$ without jeopardizing schedulability of any other task τ_j $(j \neq i)$ that has $T_j \geq T_j^S$.*

An α_i-robust period assignment allows the designer to increase the WCET of any arbitrary task such as τ_i up to $\alpha_i C_i$ while guaranteeing the schedulability for the other tasks. It means that if the implementation of one of the tasks is changed, other tasks still can use their previous set of parameters provided that the total amount of the change is smaller than $\alpha_i C_i$ for the task.

If for an α_i-robust period assignment, all α_i values in a given set $A = \{\alpha_1, \ldots, \alpha_n\}$ are equal, then we call it $\alpha-$robust instead of α_i-robust.

4. PERIOD ASSIGNMENT

In this section we explain how the set of T^S is obtained such that Definition 2 is satisfied for EDF and RM algorithms. We also discuss how to provide robustness guarantees and add new tasks to the system.

4.1 Optimal Safe Periods for EDF

The core technique which we use to find an optimal safe-period assignment for EDF is based on the work of Cervin et al., [8], where they calculate a period assignment $T^* = \{T_1^*, \ldots, T_n^*\}$ that minimizes (2) and satisfies

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1. \tag{3}$$

LEMMA 1. *(Section 3.5 from [8]): Let $(T_1^*, \ldots, T_n^*)$ be the solution of*

$$\sum_{1 \leq i \leq n} w_i T_i^* \leq \sum_{1 \leq i \leq n} w_i T_i, \tag{4}$$

for any period assignment $(T_1, \ldots, T_n)$ which satisfies (3). Then, T_i^* can be obtained by

$$T_i^* = \sqrt{\frac{C_i}{w_i}} \sum_{1 \leq l \leq n} \sqrt{w_l C_l}. \tag{5}$$

Using T^* we show how to obtain the smallest period values that minimize $\sum w_i T_i$ and are U^S-safe.

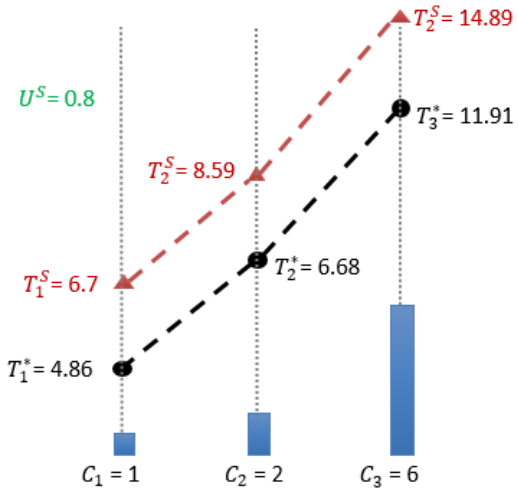

Figure 1: An example of 3 tasks with $C_1 = 1$, $C_2 = 2$, $C_3 = 6$, $w_1 = w_2 = w_3 = 1$, and $U^S = 0.8$. According to (5), we have $T_i^* = \sqrt{C_i}(\sqrt{1} + \sqrt{2} + \sqrt{6})$ and from (6) we have $T_i^S = 1.25 T_i^*$.

THEOREM 1. *Given a set of WCET and U^S, the following period assignment is an optimal U^S-safe assignment for EDF scheduling algorithm*

$$T_i^S = \frac{T_i^*}{U^S} \tag{6}$$

where T_i^* is obtained from (5).

PROOF. If T_i^* follows (5), then $U^* = 1$ because

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

$$= \frac{1}{\sum_{1 \leq l \leq n} \sqrt{w_l C_l}} \sum_{1 \leq i \leq n} \sqrt{w_i C_i} = 1 \tag{7}$$

Since T_i appears in the denominator of (3), assigning period $T_i \geq T_i^*$ to task τ_i will only reduce the utilization of τ_i, and hence, will not violate the schedulability condition of EDF. Also according to Lemma 1, T_i^* is the minimum period which can be assigned to τ_i to minimize (2). Knowing that $U^* = 1$ we have

$$\sum_{1 \leq i \leq n} \frac{C_i}{\frac{1}{U^S} T_i^*} = U^S \sum_{1 \leq i \leq n} \frac{C_i}{T_i^*} = U^S \times 1 = U^S$$

which shows that any period assignment $T_i \geq T^S$ can only have a utilization not greater than U^S. It concludes the proof. □

Fig. 1 shows the relation between T_i^* and T_i^S for an example with 3 tasks.

4.2 Robustness Guarantees for EDF

As stated earlier, components may be upgraded, repaired, or modified during the life time of a system. This modification may increase the WCET of their tasks. To avoid redoing timing analysis of the system and to keep the promised T_i^S values, in this section we show how to incorporate robustness requirements into T_i^S values such that we can keep

our promises even if the WCET of some components in the system is increased.

Suppose that a set of $A = \{\alpha_1, \ldots, \alpha_n\}$ is given that represents the upper bound of potential changes that can happen in the future for the WCETs. We try to answer the question that "if all α_i values are given, what is the safe U^S such that T^S becomes α_i-robust (while T^S is still an optimal U^S-safe set of periods)?"

THEOREM 2. *For a given set of $\{C_1, \ldots, C_n\}$ and $A = \{\alpha_1, \ldots, \alpha_n\}$, the following formula obtains U^S that guarantees an α_i-robust optimal period assignment for EDF scheduling algorithm*

$$U^S = \min\left\{\sqrt{\alpha_i}\right\}_{i=1}^{n} \times \frac{\sum_{j=1}^{n}\sqrt{\alpha_j C_j w_j}}{\sum_{j=1}^{n}\sqrt{C_j w_j}} \tag{8}$$

PROOF. Since α_i only modifies the WCET of the tasks, we can still use Lemma 1 to find the optimal period assignment for any modified WCET as follows

$$T_i^{*'} = \sqrt{\frac{\alpha_i C_i}{w_i}} \sum_{1\le l\le n} \sqrt{w_l \alpha_l C_l} \tag{9}$$

Since we want to guarantee that for any possible change (smaller than $\alpha_i C_i$) in the WCET of the tasks, $T_i^{*'}$ remains smaller than T_i^S, we will have $\forall i; T_i^{*'} \le T_i^S$. Using (6) we will have $\forall i; T_i^{*'} \le (T_i^*/U^S)$, thus

$$U^S \le \frac{T_i^{*'}}{T_i^*} \Rightarrow U^S \le \frac{\sqrt{\frac{\alpha_i C_i}{w_i}}\sum_{1\le l\le n}\sqrt{w_l\alpha_l C_l}}{\sqrt{\frac{C_i}{w_i}}\sum_{1\le l\le n}\sqrt{w_l C_l}} \tag{10}$$

By finding the minimum value of (10) among the tasks, we obtain (8). □

If a task (or component) is removed from the system, or if the system has positive slack, i.e., $U^S < 1$, it can tolerate some increases in the WCETs. Now we try to answer the question that if U^S is given, then to what extend the system can be robust.

THEOREM 3. *For a given $\{C_1, \ldots, C_n\}$, $\{T_1^S, \ldots, T_n^S\}$, and U^S, the following formula obtains the maximum value of α that guarantees an $\alpha-$robust optimal period assignment for EDF scheduling algorithm*

$$\alpha \le \frac{1}{U^S} \tag{11}$$

PROOF. Since after increasing the WCET by αC_i, the new period must still be smaller than T_i^S, $\forall i$ we have

$$T_i^{*'} \le \frac{T_i^*}{U^S} \Rightarrow \sqrt{\frac{\alpha C_i}{w_i}}\sum_{j=1}^{n}\sqrt{w_j \alpha C_j} \le \frac{T_i^*}{U^S} \Rightarrow$$

$$\alpha\sqrt{\frac{C_i}{w_i}}\sum_{j=1}^{n}\sqrt{w_j C_j} \le \frac{T_i^*}{U^S} \Rightarrow \alpha T_i^* \le \frac{T_i^*}{U^S}$$

that can be simplified to (11). □

If the whole slack-utilization (i.e., $1 - U^S$) is going to be given to one task such as τ_i, then we can use the following equations for τ_i

$$T_i^{*'} \le \frac{T_i^*}{U^S} \Rightarrow \sqrt{\frac{\alpha_i C_i}{w_i}}\Big(\sqrt{\alpha_i w_i C_i} + \sum_{1\le j\le n, j\ne i}\sqrt{w_j C_j}\Big) \le \frac{T_i^*}{U^S}$$

$$\Rightarrow \alpha_i C_i + \sqrt{\alpha_i}\left(\sqrt{\frac{C_i}{w_i}}\sum_{1\le j\le n, j\ne i}\sqrt{w_j C_j}\right) - \frac{T_i^*}{U^S} \le 0 \tag{12}$$

The inequality (12) can be represented as a quadratic equation and can be solved using the standard methods. The positive root(s) will give a candidate value for α_i. Another candidate value is obtained from the following condition that must hold for other remaining tasks in the system ($\forall j, j \ne i$):

$$T_j^{*'} \le \frac{T_j^*}{U^S} \Rightarrow \sqrt{\frac{C_j}{w_j}}\Big(\sqrt{\alpha_i w_i C_i} + \sum_{1\le l\le n, l\ne i}\sqrt{w_l C_l}\Big) \le \frac{T_j^*}{U^S}$$

$$\Rightarrow \sqrt{\alpha_i} \times \sqrt{\frac{w_i C_i C_j}{w_j}} + \left(\sqrt{\frac{C_j}{w_j}}\sum_{1\le l\le n, l\ne i}\sqrt{w_l C_l}\right) \le \frac{T_j^*}{U^S} \Rightarrow$$

$$\alpha_i \le \left(\frac{\frac{T_j^*}{U^S} - \left(\sqrt{\frac{C_j}{w_j}}\sum_{1\le l\le n, l\ne i}\sqrt{w_l C_l}\right)}{\sqrt{w_i C_i C_j / w_j}}\right)^2 \tag{13}$$

The final value of α_i is the minimum between (12) and (13) for all $j; j \ne i$.

4.3 Obtaining Safe Periods for RM

Unlike EDF, the RM may not be able to schedule a task set up to $U = 1$. However, it is known that if tasks have *harmonic* periods, i.e., each period is an integer multiple of the smaller periods, then the RM becomes optimal, namely, it can schedule any harmonic task set up to $U = 1$.

In a recent work [15], we have shown how to find harmonic periods with $U = 1$ that reduce (2). We have provided two different algorithms to construct a set of harmonic periods T_i^H that satisfy

$$\frac{\sum_{i=1}^{n} w_i T_i^H}{\sum_{i=1}^{n} w_i T_i^*} \le 2 \tag{14}$$

In other words, these algorithms have a bounded cost (error) which is not larger than two times of the optimal cost. Moreover, these algorithms have polynomial-time computational complexity. In this paper, we use DCT-based harmonic period assignment (DCT-HPA) [15] (shown in Alg. 1). Although still there is no proof that DCT-HPA provides the optimal harmonic periods (that minimize (2)), the experimental results show that it is as good as an optimal harmonic period assignment that minimizes (2) (see [15]).

DCT-HPA starts from T_1^* which is the first optimal yet non-harmonic period obtained from (5), and then tries to find closest harmonic periods that it can build for the next tasks such that for each task, the new harmonic period T_i' is not smaller than T_i^* (Lines 5 to 11). Then calculates the utilization of this assignment in Line 12. Based on this utilization, scales down all of the obtained new periods. Since they are harmonic already, if they are multiplied by a value they still remain harmonic. The scale-down process is done through Lines 13 to 15. After performing this process, the utilization of the task set becomes 1.

Since a harmonic assignment which starts from T_1^* might not necessarily be the one which leads to the minimum cost, in the next round of the for-loop, DCT-HPA tries to do the same with T_2^* and create harmonic periods from the other tasks. This process is repeated for all tasks and each time the cost is updated (Lines 17 to 20) in order to find an assignment with the smallest cost. The computational

Algorithm 1: DCT-Based Period Assignment from [15]

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

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

complexity of DCT-HPA is $O(n^2)$ because the total number of operations in Lines 6 to 11 (or Lines 13 to 15) is n.

Using DCT-HPA we obtain harmonic period assignment T_i^H which is as close as possible to the optimal period assignment T_i^* from (5). Also the utilization of this harmonic assignment is 1. To obtain T_i^S values, we can simply follow the same steps as we did for EDF.

THEOREM 4. *Given a set of WCET and U^S, the following period assignment is a U^S-safe assignment for RM scheduling algorithm*

$$T_i^S = \frac{T_i^H}{U^S} \tag{15}$$

where T_i^H is obtained from DCT-HPA.

PROOF. If T_i^H follows DCT-HPA, then $U^H = 1$. According to Baruah et al., [2], RM is sustainable towards increase in the period values. Han et al., [11], have shown that if a task set has $T_i \geq T_i^H$ where T^H is a set of harmonic periods with $U^H \leq 1$, then the task set is schedulable by RM. Consequently, if $T_i^S \leftarrow T_i^H$, the resulting set of periods is safe. Moreover, since by multiplying each of the T_i^H values by a factor $1/U^S$, still the periods remain harmonic, the resulting assignment is U^S-safe with bounded utilization U^S. □

It is important to note that our safe periods for RM are not optimal according to Definition. 2, however, their error is bounded to $2/U^S$ because the original error of the DCT-HPA is

$$\frac{\sum_{i=1}^{n} w_i T_i^H}{\sum_{i=1}^{n} w_i T_i^*} \leq 2$$

If we replace T_i^H by the U^S-safe period T_i^H/U^S (from (15)) we will have

$$\frac{\sum_{i=1}^{n} \frac{w_i}{U^S} T_i^H}{\sum_{i=1}^{n} w_i T_i^*} \leq \frac{2}{U^S} \tag{16}$$

In equation (16) if U^S is small, the difference between U^S-safe periods for RM and EDF becomes large.

4.4 Robustness Guarantee for RM

Similar to Theorem 2, for the RM scheduling algorithm we are able to obtain U^S based on a given set of α_i values such that the resulting T^S becomes α_i-robust. However, since T_i^H is obtained by our algorithm, it is hard to represent the resulting U^S as a closed form formula like what we did for EDF in (8). Instead, to obtain U^S, one can use α_i values to calculate $T_i^{*'}$ (from (9)), and sends those $T_i^{*'}$ values to DCT-HPA and gets $T_i^{H'}$ values back. Then the following inequality gives the maximum value of U^S

$$U^S \leq \min\left\{\frac{T_i^{H'}}{T_i^H}\right\}_{1\leq i\leq n} \tag{17}$$

4.5 Adding New Tasks

The spare utilization $1 - U^S$ can be used to add a set of new tasks $C^N = \{C_1^N, \ldots, C_m^N\}$ to the task set. This process can be as follows: first T_i^{N*} must be obtained from (5) solely based on the set of WCET of the new tasks, i.e., C^N. The resulting utilization will be 1. For the EDF scheduling algorithm, we scale up all periods with the same factor $1 - U^S$ as $T_i^S \leftarrow T_i^{N*}(1 - U^S)$. Although the resulting periods are safe, we have used all spare utilization in the system. It is still in the designer's hand to decide how much of the spare utilization is assigned to the new tasks.

For the RM, the solution is a bit more tricky because $T_i^{N*}(1 - U^S)$ values may not necessary be harmonic with the existing periods. An efficient solution is needed to incorporate them with the existing set of harmonic T^S periods. This part remains as one of our future work.

5. EXPERIMENTAL RESULTS

To evaluate our solutions, we perform two experiments; one based on the number of tasks and the other based on the range from which C_i values are selected. We compare period assignment for EDF (using (5), RM using DCT-HPA, and Alg. 1 from [15] which provides another set of harmonic periods. Our performance measure is the relative error (RE) of period assignment that is obtained by dividing cost of period assignment of one algorithm to the optimal cost, i.e.,

$$RE = \frac{\sum w_i T_i^{\text{algorithm}}}{\sum w_i T_i^*} \tag{18}$$

where $T_i^{\text{algorithm}}$ is the period assigned by any of the algorithms that we compare.

In the first experiment, we generate C_i with a log-uniform distribution from $[1, 500]$ and vary the number of tasks from 2 to 40. Fig. 2 shows the results of this experiment where for each value of n, 1000 random sets of WCETs are generated. As can be seen, the error of our RM solution in comparison with the optimal solution is smaller than 3%. It shows that our approximation error in (14) is way too pessimistic in comparison with the real performance of the algorithm. Moreover, our solution is salable with the increase in n.

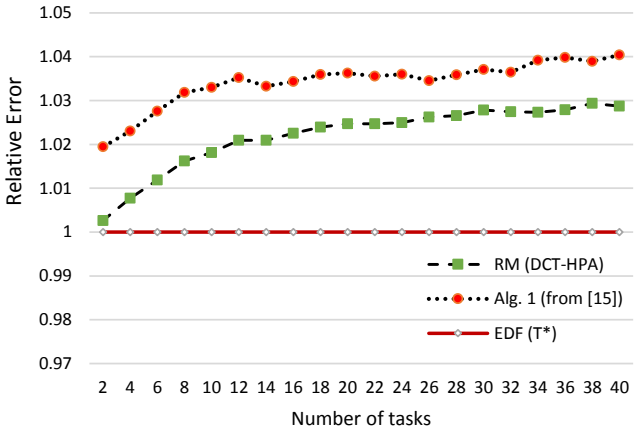

Figure 2: Effect of n on the relative error of the algorithms from (18).

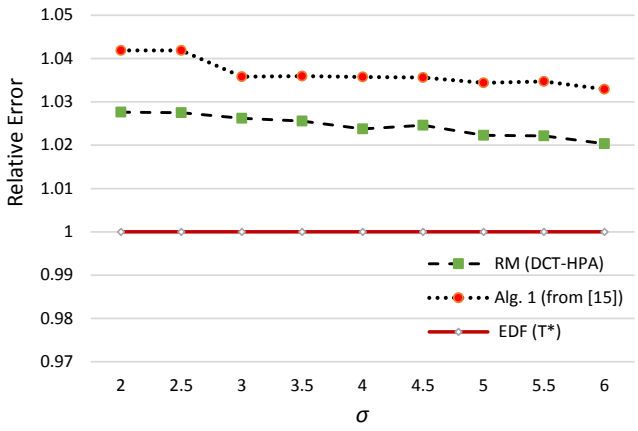

Figure 3: Effect of σ on relative error of the algorithms from (18). Note that C_i has been selected from $[1, 10^{\sigma}]$ with a log-uniform distribution.

In the second experiment, we assume $n = 20$, and then select C_i values from range $[1, 10^{\sigma}$ with log-uniform distribution. The parameter of this experiment is σ that shows how large is the range of C_i. As shown in Fig. 3, when C_i is selected from a wider range of values, the difference between C_i and C_{i+1} increases, which eventually helps our DCT-HPA solution to find a better harmonic period assignment that has smaller error.

6. RELATED WORK

Traditionally, the schedulability analysis has been done for task sets with fixed and known periods and worst-case execution times [14]. Yet, there are works such as [6,7,13,18] that consider varying period values due to the changes in the environment or occurrence of an overload. Although in these works, the robustness has been provided through run-time techniques, they are based on the assumption that the overload situation or the mode change is transient. However, in our case when the implementation of a component is modified, it will be permanent (in the sense that the system will use the new component, not the old one).

As the system enlarges, finding a set of possible configurations that guarantees all timing requirements becomes more complicated. Many researches have tried to address this problem through design space exploration [12] which provides the ability to operate on the space of design configurations, and evaluates and/or finds feasible sets of parameters. This exploration needs a mathematical model of the system which is usually integrated into one constrained optimization problem and is solved by numerical solvers. However, usually the resulting problem becomes too big to be solved. Moreover, in this approach it is not easy to add new components or modify the existing ones without changing the assigned parameters of the previous components, because then at least, the mathematical model must be rebuilt and resolved again. Besides, the space exploration tools usually use the classic schedulability tests such as response time analysis for RM or demand bound analysis for EDF. An exact solution for these tests will have considerable computational complexity since the problem of response time analysis is NP-Hard [9, 10].

In [11], two algorithms Sr and DCT are introduced to build a set of harmonic periods which are smaller than their counterpart non-harmonic periods given by the designer. If the utilization of the harmonic period set is not larger than 1, then the original task set is schedulable by RM. Our solution approach in DCT-HPA algorithm is inspired from DCT, the difference is that DCT finds a smaller harmonic period and does not re-scale it afterwards, while we build a harmonic period set which is larger than the given one, and then re-scale it.

In our recent works [16, 17], we have introduced two optimal approaches with different computational complexity for verifying the existence of a harmonic period assignment. Though they can optimally solve the *existence problem*, they cannot find an assignment with the utilization smaller than 1. In both papers, only a heuristic solution has been provided for the period assignment. In fact, we [15] have shown that the problem of finding a harmonic period assignment with the maximum feasible utilization from a given set of ranges is at least NP-Hard.

Sustainability of the schedulability tests towards certain parameters such as period, deadline, WCET, release jitter, release phase, etc. has been analyzed in [2]. This analysis tells us that the result of an schedulability test remains valid or not if at run-time, the situation becomes "better", e.g., periods increase, WCET decrease, etc. However, the work of Baruah et al., [2] does not include any analysis regarding the robustness or parameter assignment in general.

The problem of finding the maximum robustness value (or the maximum amount of execution time that can be added to a task without jeopardizing the schedulability of the task set) can be seen as an *Increased Reward with Increased Service* (IRIS) model where each task has two execution parts: mandatory and optional. In this model, executing the optional part will give a reward to the system, e.g., the reward can be a linear function of the execution of the optional part.

In this case, the minimum guarantee-able reward for the jobs of each task will show the robustness factor α_i for that task, and hence, a solution that maximizes the minimum reward will be the maximal robustness. Aydin et al., [1] have considered IRIS model and they have shown that if the reward function is linear with the execution of the optional part, then there exists a solution that maximizes the average reward of all jobs of each task [1]. However, they did not discuss the case where the goal is to maximize the minimum reward of each job. Thus, their solution cannot be used to guarantee the robustness.

Seto et al. [19] have tried to find all RM-schedulable sets of periods that are smaller than a given set of feasible periods. They first formulate the problem as an integer linear programming problem, and then provided an algorithm to find all those schedulable sets of periods. Both of these solutions have significant computational complexity. Besides, unlike our approach, they start with a feasible set of periods and limit the period space to that range, while we try to maximize the designer's choices by finding the *smallest* schedulable set of periods. Later, Bini and Di Natale [4] have extended the optimization problem in [19] to find the optimal periods that maximize an application-related goal function. They have provided an exact solution based on the branch and bound that searches all possible choices for the periods. Although using a goal function such as equation (2) we can use their approach to find the smallest schedulable periods for the RM, the computational complexity of the solution is significantly large, and can become intractable if the final priority ordering is not given (i.e., it will be $n!$).

Sensitivity analysis of fixed-priority scheduling and EDF is done in several works [3, 5]. For fixed-priority scheduling, the sensitivity analysis determines to what extent any of the parameters (or a combination of parameters) can be increased while the schedulability is preserved or can be decreased to make the system schedulable [5]. It can also be used to provide the range of feasible periods for which the schedulability is preserved. Despite being more accurate and providing solutions for much more cases (and situations) than our paper, these approaches are computationally expensive because many equations must be built from the current system parameters, and solved based on what the designer wants to verify. If the system has a large number of functionalities (or tasks), the problem becomes intractable. Moreover, this approach is passive towards guaranteeing robustness, namely, it tells the designer how much space is available for certain changes, but will not answer the reverse question which is "how much spare capacity is needed in order to accept certain amount of changes in the future without modifying parameters of unchanged components". It does not tell the designer how to assign the parameters cautiously such that certain levels of changes can be handled without modifying the other parameters. Moreover, our objective function is different from [5], i.e., we try to find the smallest set of periods that satisfy the schedulability. In this sense, our approach can be used one step earlier in the design, when the designer has not yet decided the periods, or there is nothing to begin with. Using the set of periods that we provide, the designer is able to apply any optimization criteria of his choice to optimize the period of the tasks while having a simplified schedulability test which is "any assigned period must not be smaller than T_i^S" rather than having a response time analysis problem as a part of the optimization constraints.

7. CONCLUSION

In this paper we have focused on the problem of assigning periods to a set of given WCETs. In particular, our solution decouples the schedulability analysis from the period assignment by finding a set of *safe periods*. Our solution for EDF is optimal, meaning that it finds the smallest set of periods that are as close as possible to the given set of WCETs, while our solution for RM is a polynomial-time approximation algorithm with bounded error 2. Moreover, we have shown how to calculate the safe periods such that the task set becomes α_i-robust, meaning that the new WCET of a modified component can be α_i times larger than its initial WCET. We have also derived the α_i factor according to the available slack in the system. In our experiments, the actual performance of our solution for RM was much better than its error bound, i.e., it was at most 0.03 times worse than the optimal solution (instead of 2.0). Both of our solutions for EDF and RM are polynomial-time, and they scale with the increase in the given ranges of WCETs as well as the number of tasks.

In the future, we will provide a tighter error bound for our RM solution. Moreover, we will extend the results to constrained deadline tasks, or a system that has already a set of accepted tasks. Particularly for the RM, we will provide a solution to assign harmonic periods that are consistent with the existing safe set of harmonic periods.

Acknowledgment

We would like to thank Marko Doko for his insightful comments. This work has been supported by a fellowship from Alexander von Humboldt foundation.

8. REFERENCES

[1] H. Aydin, R. Melhem, D. Mosseé, and P. Mejía-Alvarez. Optimal Reward-Based Scheduling for Periodic Real-Time Tasks. *IEEE Transactions on Computers*, 50(2):111–130, 2001.

[2] S. Baruah and A. Burns. Sustainable scheduling analysis. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 159–168, 2006.

[3] E. Bini and G. Buttazzo. The space of EDF deadlines: the exact region and a convex approximation. *Real-Time Systems*, 41(1):27–51, 2009.

[4] E. Bini and M. Di Natale. Optimal Task Rate Selection in Fixed Priority Systems. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 399–409, 2005.

[5] E. Bini, M. Di Natale, and G. Buttazzo. Sensitivity analysis for fixed-priority real-time systems. *Real-Time Systems*, 39(1-3):5–30, 2008.

[6] A. Biondi, A. Melani, M. Marinoni, M. Di Natale, and G. Buttazzo. Exact Interference of Adaptive Variable-Rate Tasks under Fixed-Priority Scheduling. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 165–174, 2014.

[7] G. Buttazzo, G. Lipari, and L. Abeni. Elastic task model for adaptive rate control. In *IEEE Real-Time Systems Symposium, (RTSS)*, pages 286–295, 1998.

[8] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Arzen. Feedback-feedforward scheduling of control tasks. *Real-Time Systems*, 23(1):25–53, 2002.

[9] F. Eisenbrand and T. Rothvoß. Static-priority real-time scheduling: Response time computation is

NP-hard. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 397–406, 2008.

[10] F. Eisenbrand and T. Rothvoß. EDF-schedulability of synchronous periodic task systems is coNP-hard. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1029–1034, Philadelphia, PA, USA, 2010.

[11] 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)*, pages 36–45, 1997.

[12] E. Kang, E. Jackson, and W. Schulte. An approach for effective design space exploration. In *Monterey Conference on Foundations of Computer Software: Modeling, Development, and Verification of Adaptive Systems*, pages 33–54. Springer-Verlag, 2011.

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

[14] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of ACM*, 20(1):46–61, 1973.

[15] M. Mohaqeqi, M. Nasri, Y. Xu, A. Cervin, and K. Arzen. On the Problem of Finding Optimal Harmonic Periods. In *International Conference on Real-Time Networks and Systems (RTNS)*, pages 171–180, 2016.

[16] M. Nasri and G. Fohler. An Efficient Method for Assigning Harmonic Periods to Hard Real-Time Tasks with Period Ranges. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 149–159, 2015.

[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)*, pages 211–220, 2014.

[18] L. Santinelli. Adaptive Schedulability Analysis . In *International Real-Time Scheduling Open Problems Seminar (RTSOPS)*, pages 21–22, 2011.

[19] D. Seto, J. P. Lehoczky, and L. Sha. Task Period Selection and Schedulability in Real-Time Systems. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 188–199, 1998.