On the Optimality of RM and EDF for Non-Preemptive Real-Time Harmonic Tasks

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

Sanjoy Baruah
Department of Computer Science, University of North Carolina, Chapel Hill, USA
baruah@cs.unc.edu

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

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

ABSTRACT

In this paper, we study non-preemptive uniprocessor real-time scheduling using the non-preemptive RM (npRM) and EDF (npEDF) scheduling algorithms. We discuss the limitations of existing studies, identifying pessimism in current schedulability analysis and inefficiencies in existing processor speedup results. Focusing on harmonic task sets, we show that even with restrictions placed on the execution times of the tasks, npRM and npEDF are not able to schedule all feasible task sets. We obtain necessary conditions for the feasibility of the harmonic tasks with arbitrary integer period ratios. Then we derive sufficient conditions for the schedulability of npRM and npEDF upon harmonic task sets. Based on these conditions, a superior speedup factor which guarantees the schedulability in cases where there are fewer restrictions on the execution times is derived. Results from simulation experiments show an average speedup factor three times less than the only existing feasible method to obtain speedup factor.

1. INTRODUCTION

In many real-time systems having interactions with the environment or communications through a shared medium, preemption is either impossible or prohibitively expensive [16]. In control applications, the delay between input and output operations (I/O delay) and its jitter are minimized using non-preemptive execution [10]. This type of execution, disturbs locality of program data in caches and avoids run-time overheads due to context switches caused by preemptions [4]. Furthermore, it enhances the evaluation of the worst-case execution time (WCET) of the tasks [29]. Using non-preemptive execution it is easy to guarantee mutual exclusion and protect shared resources among the tasks. Because of these reasons, non-preemptive execution has been used by many avionics, robotics, and cell phone applications [28, 27, 1].

RTNS 2014 , October 8 - 10 2014, Versailles, France

http://dx.doi.org/10.1145/2659787.2659806.

It has been shown that non-preemptive scheduling of real-time periodic tasks is an NP-Hard problem [18]. In 1980, an exact schedulability test for non-preemptive EDF (npEDF) has been presented by [19] which has considerable computational complexity. Jeffay et al. [18] have derived necessary and sufficient conditions for schedulability of npEDF for periodic tasks with arbitrary release offsets. According to this work, npEDF is optimal in the class of non-preemptive work-conserving scheduling algorithms. Later, sufficient conditions for schedulability of periodic tasks scheduled by non-preemptive rate-monotonic (npRM) [2, 27] have been investigated. Also in [15], schedulability of non-preemptive sporadic tasks have been studied. An schedulability analysis was presented in [24] for non-preemptive scheduling of strictly periodic tasks (those where it is required that successive jobs of a task begin execution exactly the period of the task apart).

There have been attempts to find efficient heuristics for the problem of non-preemptive scheduling. For example, clairvoyant EDF (cEDF) [13] has been proposed for firm real-time tasks. It uses a form of look ahead to determine the appropriate time to insert an idle interval before starting the execution of an urgent task. Li et al. [22] have introduced group-based EDF (gEDF) for soft real-time systems. In gEDF, tasks are grouped based on the closeness of their deadlines, and then a task with the smallest execution time among the tasks of the group with the earliest deadline is executed. However, these heuristics cannot guarantee hard deadlines.

In this paper, we show that previous scheduling algorithms and schedulability tests might be too conservative especially when they are applied on harmonic task sets: those in which each period is an integer divisor of the larger periods. These task sets have been used in a large spectrum of applications such as avionics, submarines, and robotics [8, 5, 21, 3] as well as control systems with nested feedback loops [14]. Accordingly, there have been efforts to convert peri-

odic tasks into harmonic ones [20, 17], which have also been employed in dwell tasks of Radar systems [30]. In our recent work [25], we have introduced a framework to construct customized harmonic periods provided that a valid period range is given for each task. However, non-preemptive scheduling of harmonic tasks is also an NP-Hard problem [11].

In [12], an optimal non-preemptive scheduling algorithm has been introduced for harmonic tasks with constant period ratio where each task period is K times the period of the task with the next-smallest period, for some integer constant $K \geq 3$. Later, Cai et al. [11] showed that this problem is NP-hard when the period ratios are arbitrary integer values. They introduced an optimal scheduling algorithm for binary harmonic tasks — those where each period is exactly twice the immediate smaller period. The time complexity of this algorithm is $O(2^n)$, where n is the number of tasks.

In recent work [26], we introduced Precautious-RM, an online non-preemptive scheduling algorithm with linear-time computational complexity, to efficiently schedule harmonic real-time tasks. We showed that Precautious-RM is optimal in three cases; binary tasks defined in [11], tasks with constant period ratio defined in [12], and a more general harmonic task with arbitrary integer period ratio satisfying specified relation between the periods (this relation is defined in Section 4). The focus of the current paper is on the systems which are running npRM or npEDF and they cannot apply other scheduling algorithms such as Precautious-RM.

In this paper we discuss the optimality of npRM and npEDF for harmonic task sets, and derive sufficient conditions for schedulability using these algorithms. First we show it is impossible to have a utilization-based test which guarantees non-preemptive feasibility. Next we show that the necessary and sufficient conditions for the schedulability of npEDF presented in [18] are too conservative in the context of harmonic task sets, and they have significant computational complexity. We show that npEDF and npRM have identical behavior upon harmonic task sets if the period is used as the tie breaker for jobs with equal deadline. However, we prove that even with a specified limitation on the execution time of the tasks, these algorithms are not optimal for general harmonic task sets with arbitrary integer period ratios. Moreover, we show that the existing method [31] to obtain speedup factor for guaranteeing schedulability of non-preemptive tasks is 8 for harmonic task sets which is inefficient in many cases. We further derive sufficient conditions of schedulability of npRM and npEDF for harmonic tasks with arbitrary integer period ratios. This test has two conditions; one on the length of execution times, another on the structure of the periods, i.e., the relation between period ratios. Then we show that effective speedup factor for a task set satisfying the structural constraint, is 2 when tasks have their maximum feasible execution times, i.e., the execution time of each task is two times the slack of the task with the smallest period.

According to the best of authors' knowledge, it is the first work on the optimality of npRM and npEDF for non-preemptive harmonic tasks which can also cover task sets with utilization 1 and period ratio greater than or equal to 1. Besides, our speedup factor formulation enables npEDF to execute tasks with arbitrary yet feasible execution times as long as the structural constraint holds for the periods.

The reminder of the paper is organized as follows; Section 2 introduces the system model. Then in Section 3 we show negative results on the schedulability of npRM and npEDF. Section 4 presents sufficient conditions of the optimality of npRM and npEDF. It is followed by deriving a speedup factor to relax the constraint of the execution time of the tasks. In Section 6 we present experimental results and finally the paper is concluded in Section 7.

2. SYSTEM MODEL

We consider a single processor system and hard real-time non-preemptible task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ with n independent tasks. Each task is identified by $\tau_i : (c_i, T_i)$, where c_i is the worst-case execution time (WCET), and T_i is the period of task τ_i. Tasks are periodic – not sporadic – and have no release offset, and implicit deadlines. This task model has been considered in many of control systems and multimedia applications.

We assume harmonic task sets, where each period is an integer divisor of larger periods. Tasks are assumed indexed according to their period length, i.e., $T_1 \leq T_2 \leq \ldots \leq T_n$. The period ratio is denoted by $k_i \in \mathbb{N}^+$ where $k_i = T_i/T_{i-1}$ for $1 < i \leq n$. Since each period divides smaller periods and there is no release offset, T_n is the hyperperiod of the task set. The utilization of the task set is calculated as $U = \sum_{i=1}^{n} \frac{c_i}{T_i}$.

3. NEGATIVE RESULTS FOR NON-PREEMPTIVE SCHEDULING

In this section we discuss negative results on the schedulability of non-preemptive periodic tasks with the focus on harmonic task sets.

3.1 Impossibility of Finding a Utilization-Based Test

There have been studies on deriving utilization-based test for preemptive scheduling algorithms such as EDF and RM [23, 7]. However, by the following theorem we show that in non-preemptive systems, even with utilization close to zero we may have infeasible task sets.

THEOREM 1. *Utilization of a non-preemptive task set which cannot be scheduled by any clairvoyant scheduling algorithm, can be close to zero.*

PROOF. To prove the claim we use a counter example with two periodic tasks, $\tau_1 : (\epsilon, 1)$ and $\tau_2 : (2, 1/\epsilon)$ where $\epsilon \in \mathbb{R}^+$ is an arbitrary value close to 0. When ϵ tends to 0, $u_1 \approx 0$ and $u_2 \approx 0$ which means that $U \approx 0$. Although utilization of this task set is close to 0, it is infeasible for any non-preemptive scheduling algorithm. The reason is that if at time t, task τ_1 is released and it finishes its execution until $t + c_1$, and then τ_2 enters the processor, it will not leave the processor until $t + c_1 + 2T_1$, while the next instance of τ_1 will be released at $t + T_1$ and its deadline is at $t + 2T_1$. □

Theorem 1 shows that sum of the utilization of the tasks cannot be used to construct a utilization-based test. Next we show that it is impossible to have such test which considers a set of utilization values and successfully distinguishes schedulable task sets from non-schedulable ones.

THEOREM 2. *Given set $U = \{u_1, u_2, \ldots, u_n\}$ of utilizations for a non-preemptive task set, it is impossible to find any relation between utilizations such that if it holds, schedulability of the task set is guaranteed by any scheduling algorithm.*

PROOF. To prove this claim for any given set of utilizations, we introduce task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ with the same number of tasks. Then we show this task set cannot be scheduled by any non-preemptive scheduling algorithm.

Considering an arbitrary value for the period of task τ_1, we define the parameters of the rest of the tasks as $T_1 = T_2 = \ldots = T_{n-1}$, and $c_i = u_i T_i$. We assign $c_n = 2T_1 - 2c_r + \epsilon$ for a very small positive value ϵ where $c_r = \sum_{i=1}^{n-1} c_i$. Then we set $T_n = c_n/u_n$, thus this task has utilization u_n. It is worth mentioning that because the utilization of a feasible task set cannot be more than 1, and all tasks τ_1 to τ_{n-1} have the same period T_1, sum of their execution time, i.e., c_r, has to be smaller than T_1. Consequently, c_n is always positive.

Assume the last release of tasks τ_i, $1 \leq i \leq n-1$ was at time t. Since there is no release offset, all of these tasks will be released synchronously. In the first case consider that they enter the processor before τ_n, thus, the earliest start time for τ_n would be at $t + c_r$ and task τ_n leaves the processor at $t + c_r + 2T_1 - 2c_r + \epsilon$ which is equal to $t + 2T_1 - c_r + \epsilon$. On the other hand, next release of other tasks is at $t + T_1$ and their deadline is at $t + 2T_1$, however, since they have to be executed before their deadlines, their latest start-time is $t + 2T_1 - c_r$. Regardless of the order of execution of the tasks in set $\{\tau_1, \tau_2, \ldots, \tau_{n-1}\}$, one of them will miss its deadline because it will be finished ϵ unit of time after its deadline. In the second case it is assumed that τ_n starts its execution after $x < c_r$ unit of execution times of the other tasks. Hence, to be able to successfully schedule these tasks, $c_r - x$ units of the processor time is needed before their deadline at $t + T_1$. Thus, task τ_n has to be finished before $t' = t + T_1 - (c_r - x)$. However, it is finished at $t + x + (2T_1 - 2c_r + \epsilon)$ which is after t' since

$$t + x + 2T_1 - 2c_r + \epsilon > t + T_1 - (c_r - x) \Leftrightarrow T_1 - c_r + \epsilon > 0 \quad (1)$$

As a result, some of those tasks will miss their deadline. □

3.2 Pessimism in the Existing Schedulability Tests

Jeffay et al., [18] have presented one of the pioneering works on the schedulability of non-preemptive hard real-time tasks using npEDF. They have derived necessary and sufficient conditions for the schedulability of general periodic task sets with implicit deadlines and arbitrary (unknown) release offsets scheduled by npEDF. Also they have shown that npEDF is optimal in the class of work-conserving scheduling algorithms for task sets consistent with the following theorem [18].

THEOREM 3. *Task set τ with periodic tasks sorted in non-decreasing order of their periods is schedulable if $U \leq 1$ and $\forall i, 1 < i \leq n; \forall L, T_1 < L < T_i$:*

$$L \geq c_i + \sum_{j=1}^{i-1} \lfloor \frac{L-1}{T_j} \rfloor c_j \quad (2)$$

However, these conditions are pessimistic rather than exact for task sets with specified or zero release offsets; this is shown in the following theorem.

THEOREM 4. *Theorem 3 only presents sufficient conditions for the schedulability of a task set with specified release offset r_i for each task τ_i.*

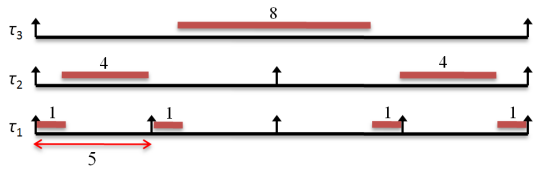

Figure 1: A counter example for necessity of the schedulability conditions presented in [18] for task sets with no release offsets

Figure 2: An example of anomaly which happens in npRM or npEDF. The schedulable task set of Figure 1 became unschedulable when the execution time of task τ_2 is reduced to 3

PROOF. The proof is based on a counter example shown in Figure 1. This example illustrates a feasible npEDF schedule for task set $\tau = \{\tau_1, \tau_2, \tau_3\}$, where $\tau_1 : (1, 5)$, $\tau_2 : (4, 10)$, $\tau_3 : (8, 20)$, and $r_i = 0$, $1 \leq i \leq n$. However, this task set will be rejected by Theorem 3, since for $i = 3$ and $L = 11$, Condition (2) is false since 11 is not greater than or equal to $8 + 1 \times \lfloor \frac{11-1}{5} \rfloor + 4 \times \lfloor \frac{11-1}{10} \rfloor$. □

Another limitation in the schedulability test of Theorem 3 is that it has pseudo-polynomial time computational complexity since it depends on the number of integer values between T_1 and T_n.

3.3 Non-Preemptive RM Is Not Optimal in General Harmonic Tasks with Limited Execution Times

Non-preemptive EDF and non-preemptive RM both have timing anomalies when they are applied to periodic task sets [9]. Figure 2 shows that the task set of Figure 1, is not schedulable by either RM or EDF if we reduce the execution time of task τ_2 from 4 to 3. In many cases, actual execution time is smaller than the worst-case execution time, hence, in practice, this type of anomaly in npEDF and npRM may happen frequently. It is one of the important drawbacks of these two algorithms when they are applied to general periodic task sets.

In this paper, we focus on harmonic task sets. With a small assumption, npRM and npEDF produce identical non-preemptive schedule in these task sets. This identical behavior is proven in the following theorem.

THEOREM 5. *Given harmonic task set τ with no release offset, the schedules generated by npRM and npEDF are identical if in npEDF we assume for two jobs with the same deadline, the one with the smaller period has the higher priority.*

PROOF. The proof is based on the fact that if the harmonic tasks have no release offset, task τ_i has a deadline which is greater than or equal to any other task τ_j, $1 \leq j <$

$i \leq n$. In other words, at any time t, the order of the periods is the same as the order of the earliness of the deadlines. In harmonic task set defined in Section 2, the order of the tasks is the same as the order of their priority in npRM because they have been sorted by the periods. Thus, to prove the theorem, we have to show that the following situation never happens in a harmonic task set: for two tasks τ_i and τ_j, $1 \leq i < j \leq n$, if τ_i is released before τ_j, deadline of τ_i is greater than or equal to deadline of τ_j. More formally, it will never happen that $r_i \leq r_j$ and $r_i + T_i > r_j + T_j$ where r_i is the release time of an instance of task τ_i. In the first step we create a formula for the relation between periods of the tasks as:

$$T_j = T_i \prod_{z=i-1}^{j} k_z, \text{ for } i < j \tag{3}$$

where k_z is the period ratio defined in Section 2. Assume for two tasks τ_i and τ_j, $i < j$, $r_i \leq r_j$ and $r_i + T_i > r_j + T_j$. By replacing T_j according to (3) we have

$$r_i + T_i > r_j + T_i \prod_{x=i-1}^{j} k_x \Rightarrow T_i(1 - \prod_{x=i-1}^{j} k_x) > r_j - r_i > 0 \tag{4}$$

However, since period ratios are positive integer values, $\prod_{x=i-1}^{j} k_x \geq 1$, hence, $1 - \prod_{x=i-1}^{j} k_x \leq 0$ which is in contradiction with (4). Consequently, such situations never happen and in all cases, the priorities assigned by RM are identical with those of EDF. □

As a direct consequence of Theorem 5, all theoretical results in this paper about harmonic task sets are applicable to both npRM and npEDF.

As mentioned before, Cai et al. [11] have derived necessary and sufficient conditions of schedulability of a ***binary*** harmonic task set where $k_i = 2$, $1 < i \leq n$, as well as a ***semi-binary*** task set with m, $0 \leq m \leq n$ binary tasks and $n - m$ tasks with $k_i \geq 3$, $m < i \leq n$. Previously, Deogun et al. [12] had derived necessary and sufficient conditions of schedulability of a task set with $c_i \leq 2(T_1 - c_1)$ and constant period ratio greater than 2. Hereafter, we call this task set as ***ternary*** task set. Figure 1 shows an example of a binary task set.

The scheduling algorithm presented by Cai et al. [11] is offline, has exponential computational complexity, and produces a scheduling table with exponentially many entries. To overcome these limitations, in our recent work [26] we have introduced Precautious-RM, which is able to schedule binary, ternary, and semi-binary harmonic task sets [12, 11] with linear time complexity. It also guarantees the schedulability of a more general task set with arbitrary integer period ratios $k_i \geq 1$, $1 < i \leq n$. However, many existing systems use npRM and npEDF. Accordingly, our goal is to derive efficient sufficient conditions for schedulability of npRM (and npEDF) in harmonic task sets. The following theorem shows two necessary conditions for non-preemptive scheduling of any arbitrary harmonic task set with no release offset.

THEOREM 6. *The following conditions are necessary for the schedulability of harmonic task set τ with no release offset and arbitrary integer period ratio $k_i \geq 1$, $1 < i \leq n$.*

$$U \leq 1 \tag{5}$$

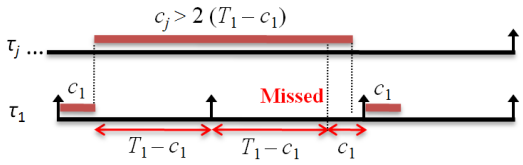

Figure 3: A counter example which shows why in harmonic task sets, maximum execution time of low priority tasks is bounded to $2(T_1 - c_1)$

$$c_i \leq 2(T_1 - c_1) \tag{6}$$

PROOF. Necessity of Condition (5) has been already proven in [18]. To validate Condition (6), we consider two cases regarding the value of k_2. If $k_2 = 1$, we have $T_2 = T_1$. Since according to (5), $U \leq 1$, we have $c_1 + c_2 \leq T_1$, because otherwise the utilization of these two tasks becomes greater than 1. Also for the same reason we have $c_1 \leq T_1$. If we add these two inequalities we will have $c_1 + c_1 + c_2 \leq T_1 + T_1$, and hence, $c_2 \leq 2(T_1 - c_1)$.

Now assume that $k_2 \geq 2$. If Condition (6) does not hold, it means that there is task τ_j, $1 < j \leq n$ with execution time $c_j = 2(T_1 - c_1) + \epsilon$ and $\epsilon > 0$. Because the task set is harmonic, this task is released at time t synchronously with task τ_1. First assume that this task enters the processor before task T_1. As a result, it will be finished at $t + 2(T_1 - c_1) + \epsilon$. The latest feasible start-time for τ_1 is $t + T_1 - c_1$, however, because of the blocking caused by τ_j, task τ_1 cannot enter the processor and misses its deadline. On the other hand, in the best case where τ_j enters the processor right after τ_1, it will leave the processor at $t + c_1 + 2(T_1 - c_1) + \epsilon$ which is greater than the latest start-time of the next instance of T_1, i.e., $t + 2T_1 - c_1$. (This situation is shown in Figure 3.) As a result, no task may have execution time greater than $2(T_1 - c_1)$. □

As shown by the example in Figure 1 and Figure 2, we can deduce that deadline miss happens when a task with execution time greater than the slack of τ_1, starts its execution just before the release of task τ_1. Consequently, τ_1 cannot find a chance to be scheduled due to the blocking caused by a low priority task. An intuitive idea to find sufficient condition for schedulability of npRM and npEDF, is to limit the execution time of the tasks to the slack of the task with the smallest period, i.e., $c_i \leq T_1 - c_1$ instead of $c_i \leq 2(T_1 - c_1)$. However, as shown by the following theorem, even with this limitation on the execution times, still npRM and npEDF are not optimal.

THEOREM 7. *npRM and npEDF are not optimal for scheduling harmonic task sets having no release offset, arbitrary integer period ratio, $U \leq 1$, and $c_i \leq T_1 - c_1$ for $1 < i \leq n$.*

PROOF. The proof is by a counter example. Consider the task set $\tau = \{\tau_1, \tau_2, \tau_3\}$ where $\tau_1 : (10, 40)$, $\tau_2 : (29, 40)$, and $\tau_3 : (30, 1200)$. Utilization of this task set is 1, and we have $c_i < 40 - 10$, $1 < i \leq 3$. However, npRM (and npEDF) produces the schedule presented in Figure 4 which is not feasible for the second instance of task τ_2. □

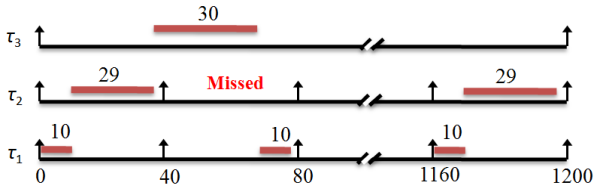

Figure 4: An schedule produced by npRM for 3 tasks $\tau_1 : (10, 40)$, $\tau_2 : (29, 40)$, and $\tau_3 : (30, 1200)$

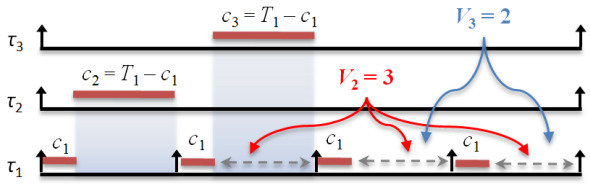

Figure 5: Vacant intervals for a task set with 3 tasks

3.4 Inefficiency of a Recent Speedup-Factor Approach

In recent work [31] it was shown that to be able to schedule any non-preemptive periodic task set that is feasible upon a unit-speed processor, with npEDF to meet all deadlines it is sufficient to use a processor with speed S, where S is obtained by the following theorem.

THEOREM 8. *The speed S that guarantees the feasibility of a non-preemptive execution of all the tasks in the task set is upper bounded by*

$$S \leq \frac{4C_{max}}{D_{min}} \tag{7}$$

where $C_{max} = \max\{c_i\}$, and $D_{min} = \min\{T_i\}$, $1 \leq i \leq n$.

Although the approach taken in this work is interesting and it can be applied on many general periodic task sets, the speed bound so obtained is very pessimistic for harmonic task sets. In the following theorem we show that maximum speedup required for npEDF to schedule an arbitrary harmonic task set is bounded by 8.

THEOREM 9. *The speed S that guarantees the feasibility of a non-preemptive execution of a harmonic task set is upper bounded by*

$$S \leq \begin{cases} 8 - 8u_1 & u_1 \leq 2/3 \\ 4u_1 & otherwise \end{cases} \tag{8}$$

PROOF. The proof is based on the necessary conditions for schedulability of harmonic tasks presented in Theorem 6. According to those conditions, $c_i \leq 2(T_1 - c_1)$ which means that $C_{max} = \max\{2(T_1 - c_1), c_1\}$. Besides, in our harmonic task set, T_1 is the minimum period. If $u_1 \leq 2/3$, c_1 will be smaller than $2(T_1 - c_1)$ because

$$u_1 \leq 2/3 \Leftrightarrow 3c_1 \leq 2T_1 \Leftrightarrow c_1 \leq 2T_1 - 2c_1 \tag{9}$$

Thus, we have

$$S \leq 4 \times \frac{2(T_1 - c_1)}{T_1} \Rightarrow S \leq 8 - 8u_1 \tag{10}$$

If u_1 tends to 0, the resulting speedup tends to 8. In the second case where $u_1 > 2/3$, we have

$$S \leq 4 \times \frac{c_1}{T_1} \Rightarrow S \leq 4u_1 \tag{11}$$

which means that S is bounded by 4 when u_1 tends to 1. □

Theorem 9 states that the task set can be feasibly scheduled by npEDF on a processor which is 8 times faster than the processor in which the timing behavior of the tasks has been measured. Since maximum utilization of a task set satisfying Theorem 6 is 1 as shown by the example in Figure 1, if we apply the speedup factor (8) to a task set with utilization 1 which is consistent with Theorem 6, the final utilization will be 0.125. For example, a simple task set with two tasks $\tau_1 : (\epsilon, 5)$ and $\tau_2 : (10 - 2\epsilon, 10)$ where $0 < \epsilon < 5$ is an arbitrary value, can be feasibly scheduled by npEDF without any speedup. However, according to the aforementioned formula, the speedup factor tends to 8 and the final utilization tends to 0.125 if ϵ tends to 0.

In the next section we discuss about the cases where npRM is optimal, then in Section 5, we derive a speedup factor which is lower than the existing results for more general cases.

4. OPTIMALITY OF NPRM IN SPECIAL CASES

In this section we derive sufficient conditions for the schedulability of npRM (and npEDF) in general harmonic task sets. In recent work [26], we have derived sufficient conditions for the schedulability of Precautious-RM in general harmonic task sets where $k_i \geq 1$, $1 < i \leq n$. However, as shown by Theorem 7, even if we limit the execution time of the tasks to $T_1 - c_1$, it is not easy to find other sufficient conditions to guarantee feasibility of npRM. We start with a definition from [26].

DEFINITION 1. *A* vacant interval *is a time interval with length $T_1 - c_1$ constructed by the slack of an instance of task τ_1 when it is executed right after its release, or just before its deadline.*

In [26], the number of vacant intervals of task τ_i, $1 < i \leq n$ is denoted by $V_i \in \mathbb{N}$ and it is defined as the number of potential vacant intervals that have not been occupied by other high priority tasks τ_j, $1 \leq j < i$. Since each vacant interval appears in the slack of one instance of τ_1, V_i becomes a function of the number of releases of τ_1 during one release of τ_i. The following recursive formula counts the number of vacant intervals for the tasks.

$$V_i = \begin{cases} k_i V_{i-1} - 1, & 1 < i \leq n \\ 1, & i = 1 \end{cases} \tag{12}$$

Note that in (12) the occupied vacant positions have been omitted in V_i. Figure 5 shows vacant intervals for a task set with 3 tasks. The following theorem presents sufficient conditions for schedulability of npRM for general harmonic tasks with arbitrary period ratio.

THEOREM 10. *Given non-preemptive harmonic task set τ with $k_i \geq 1, 1 < i \leq n$, npRM and npEDF are optimal if $U \leq 1$, $c_i \leq T_1 - c_1$, and we have $V_i \geq 1$ for $1 < i < n$ and $V_n \geq 0$.*

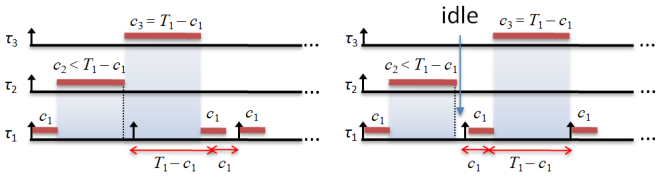

Figure 6: An example of the problem raised from having $c_i < T_1 - c_1$

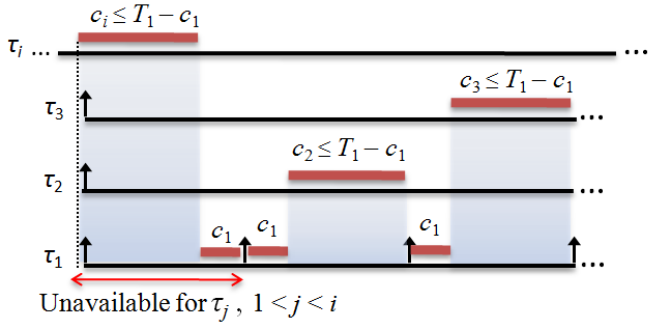

Figure 7: An example that shows how a priority inversion may happen

PROOF. In the first step we show that if those conditions hold, each task has a potential chance to be scheduled before its deadline. Since $c_i \leq T_1 - c_1$ for each i, each low priority task can fit into one vacant interval. Also V_i, $1 < i \leq n$ has to be positive, thus, there would be enough vacant intervals to schedule τ_i. Note that according to (12), one vacant interval has been reserved for the task itself. Consequently, all tasks can be scheduled before their deadlines as long as their $V_i \geq 0$.

In the second step we prove that the blocking caused by a low priority task will never cause a deadline miss for the high priority tasks. If for all tasks, we have $c_i = T_1 - c_1$, $1 < i \leq n$, no priority inversion happens since each task fits exactly into its time window (i.e., its vacant interval). As shown by Figure 6, if c_i is smaller than $T_1 - c_1$, it might happen that at the release of the next instance of τ_1, the processor is busy. If it was allowed to use idle time, such priority inversion could be handled, however, npRM and npEDF are work-conserving algorithms and they will not schedule idle times. Because of the fact that $c_i \leq T_1 - c_1$, maximum blocking time caused by a low priority task is $T_1 - c_1$ which means that it fits into the slack of the next instance of τ_1. As shown in Figure 7, when the tasks are scheduled earlier than their proper vacant interval, they may block other tasks including τ_1. Consequently, one vacant interval is wasted from all high priority tasks τ_j, $1 \leq j < i$ which might be released when the processor is blocked by the low priority task τ_i. However, this priority inversion will never cause a deadline miss since according to the conditions, each task has $V_x \geq 1$, $1 < x < n$. In other words, even if at the release of τ_j, the processor is blocked and the first vacant interval is wasted, there will be one more vacant interval for this task to be safely scheduled. □

In the next theorem we show that if $k_i \geq 2$, a task set with $c_i \leq T_1 - c_1$ is always schedulable with npRM and npEDF.

THEOREM 11. *npRM and npEDF are optimal to schedule harmonic task set τ with $k_i \geq 2$, and $c_i \leq T_1 - c_1$, $1 < i \leq n$.*

PROOF. To prove the theorem we show that if $k_i \geq 2$, V_i is always at least 1. Since each k_i is ≥ 2, formula (12) can be rewritten as

$$V_i \geq \begin{cases} 2V_{i-1} - 1, & 1 < i \leq n \\ 1, & i = 1 \end{cases} \tag{13}$$

Starting from $V_1 \geq 1$, we have $V_2 \geq 1$, $V_3 \geq 1$, $\cdots$, in each step we see V_i is at least 1. Moreover, the utilization is less than 1 because V_i values count the number of available slacks each with length $T_1 - c_1$. Consequently, V_n shows the number of available vacant intervals in the hyperperiod. Having positive $V_n \geq 0$, the amount of workload per one hyperperiod is less than 1. Consequently, all of the conditions of Theorem 10 are satisfied. □

A direct result of Theorem 11 is that npRM and npEDF are optimal for binary, semi-binary [11], ternary [12], and also general harmonic task sets [26], if $c_i \leq T_1 - c_1$ and $V_i \geq 1$, $1 < i \leq n$.

It is worth mentioning that schedulable binary, semi-binary, and ternary task sets can have utilization close to 1 if the conditions of Theorem 10 hold. For example, harmonic tasks $\tau_1 : (1, 5)$, $\tau_2 : (4, 10)$, and $\tau_3 : (4, 10)$ have $U = 1$, and they satisfy Theorem 10 since $V_2 = 1$, $V_3 = 0$, and $c_i \leq 4$. Also binary tasks $\tau_1 : (5 - \epsilon, 5)$, $\tau_2 : (\epsilon, 10)$, and $\tau_3 : (\epsilon, 20)$ have $U = 1$ for a small positive value ϵ close to 0. This task set has $V_2 = 1$, $V_3 = 1$, and $c_i \leq \epsilon$.

An important property of the presented schedulability conditions in Theorem 10 and 11 is that the schedulability is preserved even if the actual execution time of the tasks is smaller than their WCET at run-time. The reason is that those conditions limit the upper bound of WCET to be smaller than $T_1 - c_1$, which means that as long as those conditions are not violated at run-time, there will be no timing anomaly in npEDF or npRM.

5. SPEEDUP FACTOR FOR OPTIMAL NPRM SCHEDULING OF FEASIBLE TASK SETS

In this section we derive the speedup factor which can be used to schedule harmonic task sets using npRM and npEDF in feasible task sets defined by Theorem 10 with execution times bounded to $2(T_1 - c_1)$. We show that our speedup factor is much tighter than the recent results of [31].

As shown by Theorem 10, if the execution time of each task is smaller than or equal to $T_1 - c_1$, the task can be scheduled feasibly by npRM providing $V_i \geq 1$, $1 < i < n$. However, as shown by (12), V_i is a parameter of the structure of the periods, i.e., the period ratios. It means that if the relation between periods follows a special rule, the only limiting condition will be the execution times. This is the notion behind the upper bound on speedup factor derived in the following theorem for task sets with the appropriate vacant interval structure and with $c_i \leq 2(T_1 - c_1)$.

THEOREM 12. *The speed S that guarantees feasibility of the npRM and npEDF for a harmonic task set with $U \leq 1$, $c_i \leq 2(T_1 - c_1)$, $1 < i \leq n$, $V_i \geq 1$, and $V_n \geq 0$ has to be*

$$S \geq \frac{C_{max}}{T_1} + u_1 \tag{14}$$

where $C_{max} = \max\{c_i\}$, $1 \leq i \leq n$.

PROOF. By applying speedup S, we decrease the execution times of the tasks to $c'_i = c_i/S$. According to Theorem 10, the new execution times have to satisfy $c'_i \leq T_1 - c'_1$, which means that

$$c'_i \leq T_1 - c'_1 \Rightarrow \frac{c_i}{S} \leq T_1 - \frac{c_1}{S} \Rightarrow c_i \leq ST_1 - c_1 \Rightarrow$$

$$S \geq \frac{c_i + c_1}{T_1} \Rightarrow S \geq \frac{c_i}{T_1} + u_1 \Rightarrow S \geq \frac{C_{max}}{T_1} + u_1$$

□

A direct result of Theorem 12 is that our speedup factor is bounded by 2 as shown in the following theorem.

THEOREM 13. *The minimum speedup factor to guarantee schedulability of npRM and npEDF is no larger than 2 for the task sets consistent with Theorem 12.*

PROOF. By Theorem 12, any speedup S satisfying $(S \geq \frac{C_{max}}{T_1} + u_1)$ is sufficient to guarantee schedulability. Since $c_i \leq 2(T_1 - c_1)$ for all i, $C_{max} \leq 2(T_1 - c_1)$. Therefore,

$$\left(S \geq \frac{C_{max}}{T_1} + u_1\right) \Leftarrow \left(S \geq \frac{2(T_1 - c_1)}{T_1} + u_1\right) \Leftrightarrow S \geq 2 - u_1$$

which approaches 2 as u_1 tends to 0. Hence, the smallest value of S is bounded by 2. □

6. EXPERIMENTAL RESULTS

In this section, we evaluate performance of npRM against non-preemptive scheduling algorithms gEDF [22], GSSP [11], and Precautious-RM (pRM) [26]. Also we construct two non-preemptive EDF algorithms called OSP-EDF and TSP-EDF with the speedup factor. The former uses our speedup formula (OSP), i.e., formula (14), and the latter uses Thekkilakattil's speedup (TSP), i.e., formula (7). Since the algorithm presented by [12] for ternary task sets is only dedicated to the tasks with constant period ratio greater than 2, we have used GSSP which dominates Deogun's algorithm [12]. GSSP is claimed to be optimal for binary and semi-binary task sets defined in Section 3. It executes only one task between two instances of τ_1. Moreover, because of Theorem 5, outputs of npEDF are not reported since they were identical with npRM.

During the experiments we measure miss ratio (MR), i.e., the ratio of missed instances to the total number of instances of the tasks in one hyperperiod. Also for each experiment we obtain speedup factor based on our formula defined in (14) denoted by OSP, and Thekkilakattil's formula defined in (7) which is denoted by TSP value.

There are three sets of experiments based on the schedulability of the task sets. In the first set, we consider feasible tasks consistent with Theorem 10 while in the second and the third sets, general harmonic tasks are considered which might not be schedulable even with a clairvoyant non-preemptive scheduling algorithm.

Each simulation experiment averages 200 simulation runs (each including 7 randomly generated tasks). Obtained results are of 95 percent confidence level within ±0.05 confidence interval.

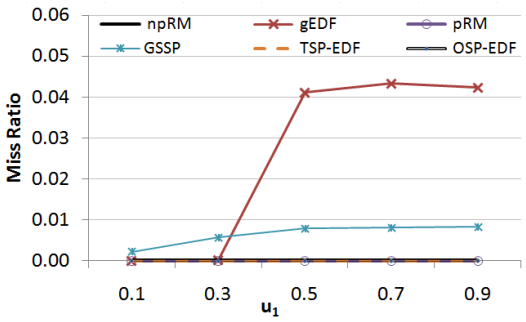

Figure 8: Miss ratio of the algorithms for different values of u_1 in feasible task sets consistent with Theorem 10. Except GSSP and gEDF, other algorithms have no miss ratio and they are overlapped

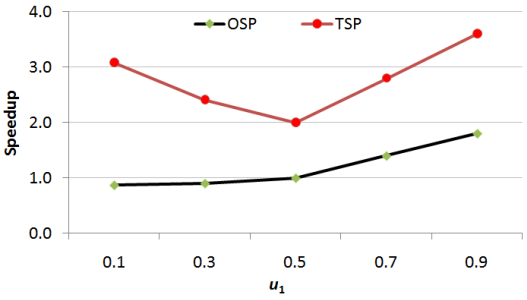

Figure 9: OSP and TSP for different values of u_1 in feasible task sets consistent with Theorem 10

6.1 Feasible Harmonic Tasks

In this section, we have considered feasible harmonic tasks satisfying conditions of Theorems 10. Parameter of the experiment is the utilization of τ_1. It varies from 0.10 to 0.90 with step 0.2. To generate the tasks, first we have chosen random values $T_1 = U[1, 10]$ with uniform chance, and $c_1 = u_1 T_1$. Then $k_i, 1 < i \leq n$ has been selected randomly from $\{1, 2, 3, \ldots, 6\}$. Other periods are then constructed as $T_i = k_i T_{i-1}$ for $1 < i \leq n$. Execution times of the tasks have been selected randomly with uniform chance as $c_i = U[0.001, T_1 - c_1]$, $1 \leq i < n$. We only accept task sets which satisfy conditions of Theorem 10, otherwise, the constructed task is not considered in the experiment.

Figure 8 and 9 illustrate values of miss ratio, OSP, and TSP for different values of u_1. The only two algorithms with miss ratio are gEDF and GSSP. GSSP has special conditions for optimality which might not be satisfied in task sets consistent with Theorem 10. Also gEDF is a greedy algorithm which aims for the execution of tasks with smaller execution times. If u_1 is higher than 0.5, c_i becomes $\leq T_1 - c_1$ which means that all other tasks will have shorter execution times than τ_1. Consequently, many of deadline misses happen for τ_1 because of the greedy choices of gEDF.

According to Figure 9, the required speedup decreases for mid-range utilizations since be the increase in u_1, value of c_i decreases and c_1 becomes the largest execution time. Hence, the speedup of formula (7) increases with the increase in the u_1 and it tends to 4 in $u_1 = 0.9$. On the other hand, when u_1 is small, other tasks might have the chance to have the largest execution time. However, since c_i is bounded to $T_1 -$

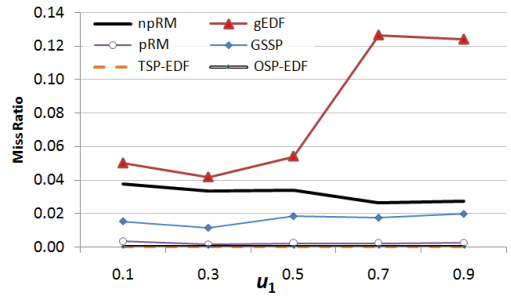

Figure 10: Miss ratio for different values of u_1 in task sets consistent with Theorem 12. OSP-EDF and TSP-EDF are overlapped

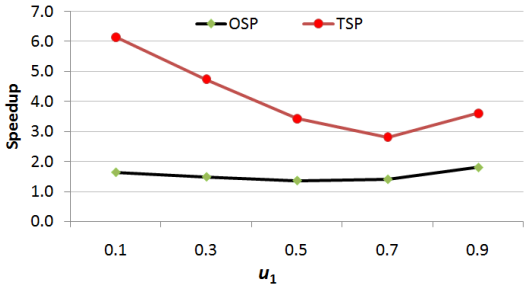

Figure 11: OSP and TSP for different values of u_1 in task sets consistent with Theorem 12

c_1, $\frac{C_{max}}{T_1}$ becomes close to 0.9 when $u_1 = 0.1$. Consequently, speedup of (7) tends to 4. Our speedup factor method has two components; u_1 and $\frac{C_{max}}{T_1}$. When u_1 is large, other tasks has smaller execution times than before, hence, the speedup factor tends to 2 since $C_{max} = c_1$ and $C_{max}/T_1 = u_1$ which lead to $S = 2u_1$. Although since npRM is optimal, there is no need to use processor speedup, still it is interesting to see that speedup factor derived in [31] is in average 2.32 times greater than our speedup factor; average TSP is 2.77 while average OSP is 1.19.

In the next step, we consider general harmonic tasks with arbitrary integer period ratio consistent with conditions of Theorem 12. The goal of this experiment is to evaluate the effectiveness of OSP in comparison with TSP. To generate the tasks we follow the steps of the previous experiment with the following modification: $c_i = U[0.001, 2(T_1 - c_1)]$, $1 \leq i < n$.

Figure 10 shows miss ratio in cases where no speedup is used. In this setup, both of OSP-EDF and TSP-EDF have had no deadline miss. According to [26], the sufficient conditions of schedulability of Precautious-RM in this case are $c_i \leq 2(T_1 - c_1)$, and $V_i \geq 0$ where $V_1 = 0.5$. The second condition is tighter than our structural condition in Theorem 12, i.e., $V_1 = 1$, hence, Precautious-RM also have some deadlines misses in the resulting task sets. Consequently, average miss ratio of Precautious-RM is 0.2%. Also as expected, the speedup methods OSP-EDF and TSP-EDF have no miss ratio while average miss ratio of normal npRM is 3%. However, as shown by Figure 11, the optimality of TSP-EDF is obtained by having average speedup factor 4.10 while our average speedup factor is 1.52 which is 2.69 times smaller than TSP.

Figure 10 also shows that gEDF is not a good scheduling algorithm for these task sets; again because of the increase in the execution time of τ_1, this task becomes the one with the largest execution time. However, since this time $c_i \leq 2(T_1 - c_1)$, the effect of the increase in c_1 does not show itself until utilization 0.7 and higher. Finally, in this figure we see GSSP has better performance than heuristic algorithms since it follows a restricted positioning for the schedule of the tasks. Accordingly, it assigns proper execution windows to τ_1, but since it will not schedule more than one task between two consecutive instances of τ_1, it still has greater deadline misses than Precautious-RM.

According to Figure 11, the highest speedup factor for TSP is required when u_1 is 0.1, because in that case, tasks are allowed to have their largest execution times. With the same reason we have mentioned in the previous experiment, when c_1 tends to T_1, it becomes the task with the largest execution time and hence, the speedup of both OSP and TSP increases.

6.2 General Harmonic Tasks with Limited Execution Times

In this experiment, we consider general harmonic tasks with arbitrary period ratio, $U \leq 1$, and $c_i \leq 2(T_1 - c_1)$ for $1 < i \leq n$. Such task set only has necessary conditions of Theorem 6. Parameter of this experiment is U which grows from 0.1 to 0.9 with step 0.2. To construct the task set first we have generated $n = 7$ utilization values with specified sum U according to uUniFast algorithm [6]. In the next step we have assigned $T_1 = U[1, 10]$ and $c_1 = u_1 T_1$ while for other periods we have followed $T_i = k_i T_{i-1}$, $1 < i \leq n$ where $k_i \in \{1, 2, 3, \ldots, 5\}$. Also the execution times of the tasks are obtained from $c_i = u_i T_i$.

Since our current formulation for the execution times, does not guarantee constraint (6), for each task with $c_i > 2(T_1 - c_1)$, we have calculated g_i as $g_i = \lfloor c_i / 2(T_1 - c_1) \rfloor$. Then we have added $g_i + 1$ more tasks to the task set; g_i tasks with execution time $2(T_1 - c_1)$ one task with execution time $c_i - 2g_i(T_1 - c_1)$. All of these tasks have the same period T_i. Consequently, no task will have execution time greater than the limit, the utilization remains constant, and the harmonic relation between periods is not destroyed. Besides, our method guarantees that each experiment has at least $n = 7$ tasks.

Figure 12 shows miss ratio for different utilizations. Since in this experiment, task set structure is not necessarily consistent with any of Theorems 10 and 12, even OSP-EDF has miss ratio greater than 0 in some cases. However, considerable difference between miss ratio of OSP-EDF and other algorithms show that with a bounded speedup less than 2, it is possible to achieve average miss ratio 0.6%, while average miss ratio of npRM is 6%. On the other hand, TSP-EDF has no miss ratio as expected since according to [31], it works for all types of periodic task sets. As shown by Figure 13 this schedulability is attained by the cost of average speedup 6.13 which is 3.70 times greater than our speedup value.

As illustrated in Figure 12, miss ratio of gEDF and npRM increases almost linearly with the increase in the utilization of the task set, while Precautious-RM is able to limit the miss ratio of the jobs to 4% at utilization 0.9. The reason is that Precautious-RM will not schedule a task if it will cause a deadline miss for the next release of τ_1. If such condition happens, it simply schedules an idle interval until the next

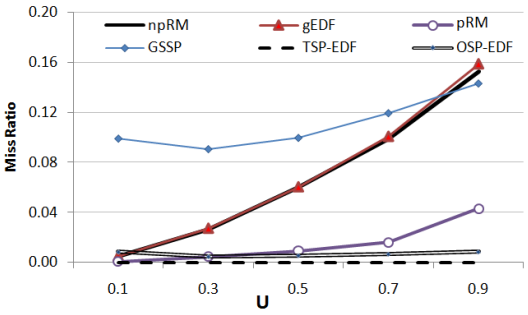

Figure 12: Miss ratio of the algorithms for task sets consistent with Theorem 6

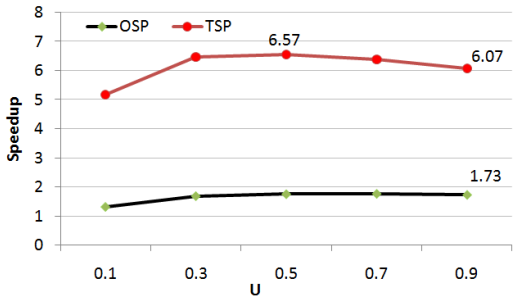

Figure 13: OSP and TSP in task sets consistent with Theorem 6

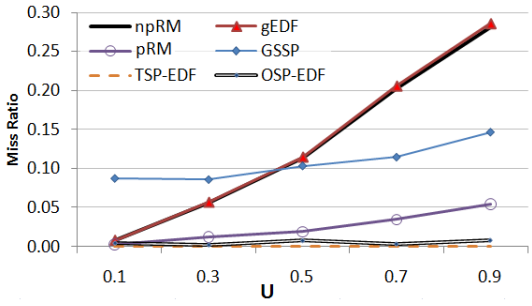

Figure 14: MR of the algorithms for arbitrary harmonic task sets. npRM and gEDF are overlapped

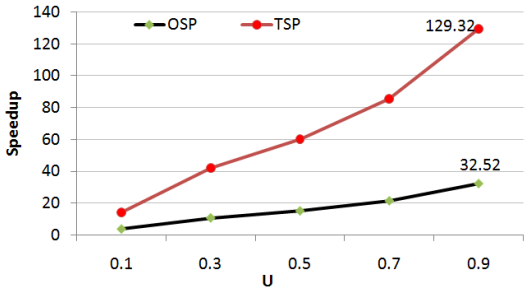

Figure 15: OSP and TSP for for arbitrary harmonic task sets

release of τ_1. After the the execution of τ_1, the task with the highest priority will take the processor and is able to use all slack of two consecutive instances of τ_1. Since in this experiment, c_i is limited to $2(T_1 - c_1)$, each task can successfully fit into that slack. In conclusion, using this small trick Precautious-RM is able to have low miss ratio without any use of processor speedup. Finally, Figure 12 shows that GSSP algorithm has higher miss ratio in utilizations smaller than 0.9 in comparison with gEDF and npRM. The reason is the strict nature of GSSP algorithm which is dedicated only for optimal cases of binary and semi-binary task sets. However, still miss ratio of GSSP is less than 14% for task sets with utilization smaller than or equal to 0.9.

In this experiment, the average value of tasks generated per task set was 20; it was 9.1 for $U = 0.1$ and 32.9 for $U = 0.9$. When the utilization is high, the chance to have longer execution times is also high which leads to have more pieces with restricted size $2(T_1 - c_1)$.

6.3 General Harmonic Tasks

In this experiment, we consider general harmonic tasks with arbitrary period ratio, $U \leq 1$, and no restriction on the execution times. We generate the tasks as said in Section 6.2 with this modification that we do not cut the execution time of the tasks into smaller pieces with limited size. Figure 14 and Figure 15 represent miss ratio and speedup values of the algorithms for different utilizations. The first noticeable fact is that gEDF and npRM have increasing miss ratio with the increasing utilization. GSSP shows to be more reliable in this case since it provides execution windows for the tasks, however, it still has about 10% miss ratio in average. Precautious-RM is a very efficient algorithm among those with no processor speedup. However, it has to be mentioned that it will not schedule a task with execution times greater than $2(T_1 - c_1)$ since such task will cause deadline miss for τ_1.

7. CONCLUSION

In this paper we discuss the optimality of non-preemptive RM and EDF for harmonic task sets. First we show that a) it is impossible to define a utilization based test to verify feasibility of a non-preemptive task set, b) current existing schedulability tests are too conservative or computationally unaffordable, c) even with the restrictions on the execution times of the tasks in a harmonic task set, still npRM and npEDF are not optimal, and d) current processor speedup factor for guaranteeing feasibility of harmonic task sets is bounded to 8 which is not so efficient. Then we derive sufficient conditions of the optimality of npRM and npEDF which will be the basis of our speedup factor formulation. The obtained speedup factor is bounded to 2 and it guarantees feasibility of the schedule for feasible harmonic tsk sets as long as some structural constraints hold for the periods of the tasks. Our experiments have shown that in average, the existing speedup factor formula is nearly 3 times higher than our formulation.

Acknowledgment

This work has been supported by the German Academic Exchange Program (DAAD) scholarship.

8. REFERENCES

[1] A. Al-Sheikh, O. Brun, P. E. Hladik, and B. J. Prabhu. A best-response algorithm for multiprocessor periodic scheduling. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 228–237, 2011.

[2] B. Andersson and E. Tovar. The utilization bound of non-preemptive rate-monotonic scheduling in Controller Area Networks is 25%. In *IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 11–18, 2009.

[3] S. Anssi, S. Kuntz, S. Gérard, and F. Terrier. On the gap between schedulability tests and an automotive task model. *Journal of Systems Architecture*, 59(6):341–350, 2013.

[4] A. Bastoni. Cache-related preemption and migration delays: Empirical approximation and impact on schedulability. In *International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)*, pages 33–44, 2010.

[5] I. Bate, A. Burns, J. McDermid, and A. Vickers. Towards a fixed priority scheduler for an aircraft application. In *Euromicro Workshop on Real-Time Systems (EWRTS)*, pages 34–39, 1996.

[6] E. Bini and G. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[7] E. Bini, G. Buttazzo, and G. Buttazzo. A hyperbolic bound for the rate monotonic algorithm. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 59–69, 2001.

[8] 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)*, pages 195–202, 1996.

[9] G. C. Buttazzo. Rate monotonic vs. edf: Judgment day. *Real-Time Systems*, 29(1):5–26, 2005.

[10] G. C. Buttazzo, M. Bertogna, and G. Yao. Limited preemptive scheduling for real-time systems: A survey. *IEEE Transactions on Industrial Informatics*, 9(1):3–15, 2013.

[11] Y. Cai and M. C. Kong. Nonpreemptive scheduling of periodic tasks in uni- and multiprocessor systems. *Algorithmica*, 15(6):572–599, 1996.

[12] J. S. Deogun and M. C. Kong. On periodic scheduling of time-critical tasks. In *IFIP World Computer Congress*, pages 791–796, 1986.

[13] C. Ekelin. Clairvoyant non-preemptive EDF scheduling. *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 23–32, 2006.

[14] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. D. Koutsoukos, and H. Wang. Feedback thermal control for real-time systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 111–120, 2010.

[15] L. George, N. Rivierre, and M. Spuri. Preemptive and non-preemptive real-time uni-processor scheduling. Technical report, INRIA Research Report nRR2966, 1996.

[16] M. Grenier and N. Navet. Fine-tuning MAC-level protocols for optimized real-time QoS. *IEEE Transaction on Industrial Informatics*, 4(1):6–15, 2008.

[17] 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.

[18] K. Jeffay, D. F. Stanat, and C. U. Martel. On non-preemptive scheduling of period and sporadic tasks. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 129–139, 1991.

[19] K. H. Kim and M. Naghibzadeh. Prevention of task overruns in real-time non-preemptive multiprogramming systems. In *International Symposium on Computer Performance Modelling, Measurement and Evaluation (PERFORMANCE)*, PERFORMANCE '80, pages 267–276, 1980.

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

[21] 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)*, pages 10–18, 2003.

[22] W. Li, K. Kavi, and R. Akl. A non-preemptive scheduling algorithm for soft real-time systems. *Computers and Electrical Engineering*, 33(1):12–29, 2007.

[23] 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.

[24] M. Marouf and Y. Sorel. Schedulability conditions for non-preemptive hard real-time tasks with strict period schedulability analysis. In *Conference on Real-Time and Network Systems (RTNS)*, pages 50–58, 2010.

[25] 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) (to appear)*, 2014.

[26] M. Nasri and M. Kargahi. Precautious-rm: a predictable non-preemptive scheduling algorithm for harmonic tasks. *Real-Time Systems*, 50(4):548–584, 2014.

[27] M. Park. Non-preemptive fixed priority scheduling of hard real-time periodic tasks. In *International Conference on Computational Science*, pages 881–888, 2007.

[28] M. Piaggio, A. Sgorbissa, and R. Zaccaria. Preemptive versus non-preemptive real-time scheduling in intelligent mobile robotics. *Journal of Experimental and Theoretical Artificial Intelligence*, 12(2):235–245, 2000.

[29] H. Ramaprasad and F. Mueller. Bounding worst-case response time for tasks with non-preemptive regions. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, page 58âĂŞ67, 2008.

[30] 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)*, pages 210–219, 2003.

[31] A. Thekkilakattil, R. Dobrin, and S. Punnekkat. Quantifying the sub-optimality of non-preemptive real-time scheduling. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 113–122, 2013.