Parameterized Schedulability Analysis on Uniform Multiprocessors

Risat Mahmud Pathan and Jan Jonsson
Chalmers University of Technology
SE-412 96, Göteborg, Sweden
{risat, janjo}@chalmers.se

Abstract

This paper addresses global rate-monotonic scheduling of implicit-deadline periodic real-time tasks on uniform multiprocessor platforms. In particular, we propose new schedulability conditions that include a set of easily computable task-set parameters for achieving better system utilization while meeting the deadlines of all the tasks. First, an individual sufficient schedulability condition is derived for each task. Then, the collection of schedulability conditions for the tasks are condensed to provide two different simple sufficient schedulability conditions for the entire task system — one for uniform multiprocessors, and one for unit-capacity multiprocessors, respectively. Finally, we show that our proposed simple rate-monotonic schedulability conditions for uniform and unit-capacity multiprocessors have higher worst-case system utilization than all other state-of-the-art simple schedulability conditions for global rate-monotonic scheduling of implicit-deadline tasks.

1 Introduction

Real-time system applications are often represented using a finite collection of independent recurring processes, for example, in control and monitoring applications. Such recurrent processes are typically modeled using the preemptive *periodic task model* in which each periodic task's relative deadline is equal to its period (*implicit-deadline*) [18]. One of the main challenges for such a task model is designing fixed-priority scheduling algorithm that ensures that all the deadlines of the tasks are met during runtime. The fixed-priority Rate-Monotonic (RM) scheduling algorithm [18] is widely used in industry because of its simplicity, flexibility and its ease of implementation [22]. Even though the RM scheduling is optimal for uniprocessor platform, its schedulability performance is poor on multiprocessor platform [13]. Our endeavor in this paper is the design and analysis of RM scheduling to achieve higher system utilization on multiprocessor platform in which the processors may have different speeds (*uniform multiprocessors*).

Real-time multiprocessor systems use either *global* scheduling or *partitioned* scheduling. In global scheduling, a task is allowed to execute on *any* processor even when it is resumed after preemption. This is done by keeping all the ready tasks in a global queue from which the highest-priority tasks are dispatched to the processors for execution. In partitioned scheduling, the task set is grouped in different task partitions during design time and each partition has a *fixed* processor onto which only the tasks of that partition are allowed to execute. Many policies for the global [2, 20, 4, 5, 8, 6, 1, 9] and partitioned [13, 21, 7, 3, 19, 17] approaches exist for fixed-priority multiprocessor scheduling. To this end, much research has focused on finding an efficient *schedulability condition* that, if satisfied, implies that all the deadlines of the tasks are met. In this paper, we present a schedulability analysis (sufficient schedulability condition) for global preemptive RM scheduling for implicit-deadline tasks on uniform multiprocessors and its specialization on unit-capacity multiprocessors.

A system consisting of multiple processors having different computing capacity (speed) is called a *uniform multiprocessor platform* [15]. On such a platform, a processor with speed s completes $s \times t$ units of execution when a task executes for t time units on the processor. If the speed of each processor is one, then it is a *unit-capacity multiprocessor platform*. The importance of multiprocessor scheduling on uniform processors has already been pointed out in [14]. In addition, we find two more reasons to study scheduling on uniform multiprocessors. First, when the frequency of different processors' clocks are lowered to save power, the schedulability of the tasks on such a platform can be analyzed using uniform multiprocessor theory. Second, it is very likely that chip multiprocessors [11] will have cores with different speeds in the future.

The quality of many multiprocessor scheduling algorithms is measured in terms of the *worst-case system utilization* (also known as *utilization bound*). The worst-case system utilization of a scheduling algorithm is used to provide a *sufficient schedulability condition* for that algorithm. This is because any set of tasks is guaranteed to be schedulable (i.e. meet all the deadlines) using a particular scheduling algorithm whenever the *total utilization* (computation load) of the task set is no larger than the worst-case system utilization. This fact motivates us to design and analyze the RM scheduling algorithm for the multiprocessor platform in order to achieve a high worst-case system utilization.

The main reason for using the total utilization of a task set in a sufficient schedulability condition is that this metric is easy to compute and also represents a simple and condensed *task-set parameter* in the schedulability condition. In this paper, we introduce three new task-set parameters (formally defined later) that can be easily computed for any task set — *the minimum period ratio*, *the maximum period ratio*, and *the sum of squares of individual task utilization*. Using these parameters, we derive a sufficient schedulability condition for global RM scheduling of implicit-deadline periodic task sets that has a higher worst-case system utilization compared to that of all other similar conditions for global RM scheduling. To this end, we make the following major contributions in this paper:

1. Using the three new task-set parameters (mentioned above), an individual sufficient RM schedulability condition for each task is derived for uniform multiprocessors (i.e. all the sufficient conditions are checked iteratively for all the tasks). These individual conditions are then condensed into two simple (i.e. non-iterative) sufficient RM schedulability conditions for the entire task set — one for uniform multiprocessors and one for unit-capacity multiprocessors.
2. To the best of our knowledge, our simple schedulability conditions for the uniform and unit-capacity multiprocessors dominate *all* such simple RM schedulability conditions proposed by others for implicit-deadline task systems[1]. However, for other simple fixed-priority schedulability conditions that our simple conditions do not dominate, it is also true that those conditions do not dominate ours either.
3. The three new task-set parameters have an important implication for the system designer in that there exist certain strategies for selecting these task-set parameters so as to achieve a good system performance.

The rest of the paper is organized as follows. First, in Section 2, other works related to multiprocessor scheduling are discussed. In Section 3, the task model used in this work is presented. The model of the multiprocessor platform and some related theory is discussed in Section 4. Then, we present our schedulability analysis for uniform and unit-capacity multiprocessors in Section 5. In Section 6, we compare the performance of our schedulability condition with other works. Finally, Section 7 concludes this paper.

[1]By domination we mean that all task sets that satisfy other schedulability conditions must also satisfy our condition, and not vice versa.

2 Related Work

Fixed-priority multiprocessor scheduling with non-zero worst-case system utilization was first proposed in [2] using a concept called *utilization separation*. The `RM-US` algorithm in [2] assigns the highest priority to the tasks having individual task's utilization greater than $\frac{m}{3m-2}$ (m is the number of processors) while the rest of the tasks are assigned priorities according to RM. It is proved that the worst-case system utilization of algorithm `RM-US` is $\frac{m^2}{3m-2} \approx 33.33\%$ on m processors (as $m \to \infty$). The worst-case system utilization of hybrid priority-assignment algorithms such as `RM-US` has been shown to be as high as approximately $37.48m \approx 37.48\%$ [20]. Hybrid priority-assignment based on the slack (i.e. task's period minus the task's execution time) of each task was proposed in [1]. The worst-case system utilization of the `SM-US` algorithm in [1] is approximately $\frac{2m}{3+\sqrt{5}} \approx 38.19\%$ for implicit deadline tasks. Recently, a new fixed-priority assignment scheme for global scheduling was proposed [12], which has been shown (using simulation) to have better performance than other fixed-priority scheduling algorithm.

Based on the minimum amount of interference in an interval that can cause a task's deadline to be missed, a fixed-priority schedulability analysis is given in [4], showing a worst-case system utilization of $\frac{m(1-u_{max})}{2} + u_{min}$ for RM scheduling on m processors, where u_{max} and u_{min} are the maximum and minimum utilization of any task, respectively. RM scheduling on uniform multiprocessors is studied in [5], and it is shown (using a schedulability analysis similar to that of [2]) that the worst-case system utilization is $\frac{m}{3}$ for RM scheduling on m unit-capacity processors.

The work in [8] derives a simple (i.e. non-iterative) sufficient RM schedulability condition that dominates the simple schedulability conditions in [2, 4, 5] for implicit deadline tasks. Using an analysis of the worst-case workload in an interval, the analysis in [8] shows that, if the total utilization of a task set is no larger than $\frac{m(1-u_{max})}{2} + u_{max}$, then all the tasks deadlines will be met during run time. The work in [8] is further extended and improved in [9], where an iterative algorithm for testing the schedulability of each task separately is proposed. Iterative sufficient scheduling conditions are also derived in [6] for fixed-priority tasks with arbitrary deadlines. Our objective in this paper is to derive simple (i.e. non-iterative) sufficient schedulability conditions, where only one condition has to be verified for the complete task set to determine the RM schedulability on a uniform or unit-capacity multiprocessor platform. Also note that we analyze the schedulability of task sets having traditional RM priority. However, our schedulability analysis can be easily extended to hybrid priority-assignment (similar to [16]) using the concept of utilization-separation.

3 Task Model and Related Definitions

In this paper, we consider the scheduling of n periodic tasks in the set $\{\tau_1, \tau_2, \ldots \tau_n\}$. Each task τ_i in set $\{\tau_1, \tau_2, \ldots \tau_n\}$ is characterized by a pair (C_i, T_i), where C_i represents the worst-case execution time (WCET) and T_i is the period (inter-arrival time) of task τ_i. Since the task set is periodic, an instance of each task, called *job*, is released at each period T_i and requires at most C_i units of execution time before the next period. Thus, the relative deadline of a task τ_i is equal to its period T_i, that is, the set $\{\tau_1, \tau_2, \ldots \tau_n\}$ is an *implicit-deadline* periodic task system.

The fixed priority of the tasks are given according to the RM discipline. In other words, the shorter the task period, the higher the task priority. Without loss of generality, we assume that task τ_i has higher priority than that of task τ_{i+1}, for all i, $1 \leq i < n$. Therefore, we have $T_i \leq T_j$ for all i and j such that $1 \leq i \leq j \leq n$.

Since the execution of a task τ_k can be interfered only by the higher-priority tasks using preemptive RM scheduling, whether task τ_k meets its deadline or not depends on the tasks in $\{\tau_1, \tau_2, \ldots \tau_k\}$ (but are completely unaffected by the tasks in $\{\tau_{k+1}, \tau_{k+2}, \ldots \tau_n\}$.) We find it useful to define the task set $\Gamma^k \overset{\text{def}}{=} \{\tau_1, \tau_2, \ldots \tau_k\}$ for $k = 1, 2 \ldots n$. Note that $\Gamma^n = \{\tau_1, \tau_2, \ldots \tau_n\}$. The collection of all the jobs of all the k tasks in set Γ^k constitute an implicit-deadline task set Γ^k. We sometimes, hereafter, also use the notation Γ^k to denote the collection of all the jobs in Γ^k.

Next, we define a couple of task-set parameters, and some important properties of these parameters, that we will use later in this paper.

We define the *load* or *utilization* of a task τ_i as $u_i = C_i/T_i$ and the *total utilization* of the task set Γ^k as $U^k = \sum_{i=1}^{k} u_i$ for $k = 1, 2 \ldots n$. Note that U^n is the total utilization of the task set Γ^n.

We define the *maximum utilization* u^k_{max} of a periodic task system Γ^k to be the largest utilization of any task in Γ^k such that $u^k_{max} \overset{\text{def}}{=} max_{\tau_i \in \Gamma^k} \quad u_i$.

Similarly, we define the *minimum utilization* $u^k_{min}$of a periodic task system Γ^k to be the smallest utilization of any task in Γ^k such that $u^k_{min} \overset{\text{def}}{=} min_{\tau_i \in \Gamma^k} \quad u_i$.

Note that, for $k \leq n$, we must have

$$u^n_{min} \leq u^k_{min} \leq u^k_{max} \leq u^n_{max} \tag{1}$$

We define the *minimum period ratio* r'_k of a task set Γ^k to be the minimum of the ratios of periods of any two tasks $\tau_i \in \Gamma^k$ and $\tau_j \in \Gamma^k$ such that $T_i \leq T_j$ and $i \neq j$. Formally,

$$r'_k \overset{\text{def}}{=} \underset{1 \leq i < j \leq k}{min} \{T_i/T_j\} \tag{2}$$

Similarly, we define the *maximum period ratio* r''_k of a task set Γ^k to be the maximum of the ratios of periods of any two tasks $\tau_i \in \Gamma^k$ and $\tau_j \in \Gamma^k$ such that $T_i \leq T_j$ and $i \neq j$. Formally,

$$r''_k \overset{\text{def}}{=} \underset{1 \leq i < j \leq k}{max} \{T_i/T_j\} \tag{3}$$

Note that, using Eq. (3) for any two tasks $\tau_i \in \Gamma^k$ and $\tau_j \in \Gamma^k$ such that $T_i \leq T_j$ and $i \neq j$, we have

$$\frac{T_i}{T_j} \leq r''_k \quad \equiv \quad \frac{T_i}{r''_k} \leq T_j \tag{4}$$

Also note that, for task systems Γ^k and Γ^n such that $k \leq n$, we also have

$$r'_n \leq r'_k \leq r''_k \leq r''_n \leq 1 \tag{5}$$

We, furthermore, define Q^k as the *sum of squares of individual task utilization* of the tasks in a task set Γ^k excluding the task with maximum utilization in Γ^k by

$$\mathrm{Q}^k \overset{\text{def}}{=} \sum_{i=1}^{k} u_i^2 - (u^k_{max})^2 = \sum_{\substack{i=1 \\ i \neq max\_index(k)}}^{k} u_i^2 \tag{6}$$

such that $max\_index(k)$ is the index of the task that has the maximum utilization in the task set Γ^k for $k = 1, 2, \ldots n$. Since $(u^k_{max})^2$ is not included in Q^k, we must have

$$\mathrm{Q}^k \leq \sum_{i=1}^{k-1} u_i^2 \tag{7}$$

And, for n, we have the following:

$$\begin{aligned} & U^n - u^n_{max} - \mathrm{Q}^n \geq 0 \qquad (8) \\ \equiv & \sum_{\substack{i=1 \\ i \neq max\_index(n)}}^{n} u_i - \mathrm{Q}^n \geq 0 \\ \equiv & \sum_{\substack{i=1 \\ i \neq max\_index(n)}}^{n} u_i - \sum_{\substack{i=1 \\ i \neq max\_index(n)}}^{n} u_i^2 \geq 0 \end{aligned}$$

$\Leftarrow$ (From Eq. (6) and the fact that $u_i^2 \leq u_i \leq 1$ for any τ_i)

From Eq. (5), we have $(r'_n \leq r''_n) \Rightarrow \left(\frac{r'_n}{r''_n} \leq 1\right)$. Therefore from Eq. (8) we have,

$$U^n - u^n_{max} - \frac{r'_n}{r''_n} \cdot \mathrm{Q}^n \geq 0 \tag{9}$$

and since $u^n_{max} \geq u^n_{min}$ from Eq. (1), we also have,

$$U^n - u^n_{min} - \frac{r'_n}{r''_n} \cdot \mathrm{Q}^n \geq 0 \tag{10}$$

4 System Model and Related Theory

The periodic task system defined in Section 3 is to be scheduled on a multiprocessor system. The following three definitions (Definition 1, 2, 3) from [5] are the key concepts, regarding scheduling on uniform multiprocessor platform, that we will use in this paper.

Definition 1 (from [5])**.** *Let π denote a uniform multiprocessor platform.*

- *The number of processors in π is denoted by $m(\pi)$*
- *For all i, $1 \leq i \leq m(\pi)$, the speed of the i^{th} fastest processor in π is denoted by $s_i(\pi)$, i.e., the speed are indexed in a non-increasing order.*
- *The total computing capacity of all the processors in π is denoted by $S(\pi) \overset{\text{def}}{=} \sum_{i=1}^{m(\pi)} s_i(\pi)$.*

Next we present two useful parameters from [5] that are defined to measure the degree by which uniform multiprocessor platform π differs from an identical or unit-capacity multiprocessor platform.

Definition 2 (λ and μ from [5])**.** *For any uniform multiprocessor platform π, we define a parameter $\underline{\lambda(\pi)}$ as follows:*

$$\lambda(\pi) \overset{\text{def}}{=} \max_{i=1}^{m(\pi)} \left\{ \frac{\sum_{j=i+1}^{m(\pi)} s_j(\pi)}{s_i(\pi)} \right\}$$

For any uniform multiprocessor platform π, we define a parameter $\underline{\mu(\pi)}$ as follows:

$$\mu(\pi) \overset{\text{def}}{=} \max_{i=1}^{m(\pi)} \left\{ \frac{\sum_{j=i}^{m(\pi)} s_j(\pi)}{s_i(\pi)} \right\}$$

If the speeds of the processors differs from each other significantly, we have $\lambda(\pi) = 0$ and $\mu(\pi) = 1$, and if we are considering identical or unit-capacity multiprocessor platform, we have $\lambda(\pi) = (m(\pi) - 1)$ and $\mu(\pi) = m(\pi)$. Note that, the relationship between $\mu(\pi)$ and $\lambda(\pi)$ is as follows.

$$\mu(\pi) = \quad 1 + \lambda(\pi) \tag{11}$$

Scheduling algorithm on uniform multiprocessor requires a stronger notion of work-conserving algorithm (defined below), called *greedy scheduling algorithms*, that is defined next according to [5].

Definition 3 (Greedy Scheduling Algorithm)**.** *A uniform multiprocessor scheduling is* ***greedy*** *if it satisfies all three of the following conditions.*

1. *It is work conserving, that is, it never idles a processor when there are jobs awaiting execution.*
2. *If it must idle some processor (fewer active jobs than processors), then it idles the slowest processor.*
3. *It always executes higher priority jobs upon faster processors. If job J_1 and J_2 are executing on the i^{th} and j^{th} processors for $i < j$, then job J_1's priority is higher than the priority of job J_2.*

In rest of the paper, we consider that the global scheduling algorithm RM is greedy. *Our objective in this paper is to analyze the RM schedulability of a task set Γ^n on a uniform multiprocessor platform π.* To characterize the amount of work done by any scheduling algorithm A on a uniform multiprocessor platform π over a time interval of length t, we have the following well-known definition of work [5].

Definition 4 ($W(A, \pi, \Gamma^k, t)$)**.** *Let any uniform multiprocessor platform π on which the task set Γ^k is to be executed using any algorithm A. For time instant $t \geq 0$, let $W(A, \pi, \Gamma^k, t)$ denote the amount of work done by algorithm A on jobs of the task set Γ^k over the interval $[0, t)$, while executing on π.*

The following Theorem 1 (proved in [14]) provides a lower bound on the amount of work completed by a greedy scheduling algorithm A on a uniform multiprocessor platform π, in terms of the amount of work completed by algorithm A_0 on a uniform multiprocessor platform π_0, when the inequality in Eq. (12) is satisfied for the platforms π and π_0. We will be using Theorem 1 later in this section.

Theorem 1 (From [14])**.** *Let π_0 and π denote uniform multiprocessor platform. Let A_0 denote any uniform multiprocessor scheduling algorithm, and A any greedy uniform multiprocessor scheduling algorithm. If the following condition is satisfied by platforms π and π_0:*

$$S(\pi) \geq S(\pi_0) + \lambda(\pi) \cdot s_1(\pi_0) \tag{12}$$

then, for any collection of jobs of task set Γ^k and any time instant $t \geq 0$,

$$W(A, \pi, \Gamma^k, t) \geq W(A_0, \pi_0, \Gamma^k, t) \tag{13}$$

It will be evident later that the schedulability analysis presented in Section 5 is based on the lower and upper bounds of the amount of work completed within a particular time interval by the greedy RM scheduling on a uniform multiprocessor platform π. Next we present the results of some previous works that estimate the lower and upper bound of the amount of work completed by the RM scheduling algorithm.

4.1 Lower Bound on Work

In this subsection we calculate the lower bound on the amount of work done within a time interval $[0, t)$ by the greedy RM scheduling on jobs of the task set Γ^k. The following Lemma 1 (proved in [5]) shows that there exists a uniform multiprocessor platform π_0 on which a task system Γ^k is schedulable.

Lemma 1 (From [5])**.** *Task system Γ^k is feasible on a uniform multiprocessor platform π_0 satisfying the conditions:*

$$S(\pi_0) = U^k \quad \textit{and} \quad s_1(\pi_0) = u^k_{max}$$

The proof of Lemma 1 is based on the construction of the uniform multiprocessor platform π_0 with k processors with speed $s_1(\pi_0)$, $s_2(\pi_0) \ldots s_k(\pi_0)$ such that $s_i(\pi_0) = u_i$ and u_i is the utilization of the task $\tau_i \in \Gamma^k$. Note that, such a platform must satisfy $S(\pi_0) = U^k$ and $s_1(\pi_0) = u^k_{max}$ for the task set Γ^k for $k = 1, 2, \ldots n$. The task set Γ^k is schedulable on the platform π_0 using the scheduling algorithm, called `OPT`, that exclusively executes the task $\tau_i \in \Gamma^k$ on the processor with speed s_i.

According to the definition of uniform multiprocessor, a task τ_i executes on processor with speed s_i for t time units completes $t \times s_i$ units of execution. Therefore, the amount of work completed within the interval $[0, t)$ by the scheduling algorithm `OPT` on the jobs of the task set Γ^k on platform π_0 is given by Eq. (14).

$$W(\mathrm{OPT}, \pi_0, \Gamma^k, t) = \sum_{i=1}^{k} t \cdot s_i = t \cdot \sum_{i=1}^{k} u_i \qquad (14)$$

Recall that we want to execute a task set on uniform multiprocessor platform π using greedy RM scheduling. If the uniform multiprocessor platforms π and π_0 satisfies the condition in Eq. (12) of Theorem 1 (where π_0 is the platform guaranteed to exist according to Lemma 1) and RM is greedy scheduling, then using inequality Eq. (13) of Theorem 1 we have

$$W(\mathrm{RM}, \pi, \Gamma^k, t) \geq W(\mathrm{OPT}, \pi_0, \Gamma^k, t) = t \cdot \sum_{i=1}^{k} u_i \qquad (15)$$

According to Eq. (15), the amount of work completed over the interval $[0, t)$ by the RM scheduling on the jobs of the task set Γ^k while executing on platform π is lower bounded by $t \cdot \sum_{i=1}^{k} u_i$ <u>if the inequality in Eq. (12) is satisfied</u> for platforms π and π_0. It is worth mentioning at this point that Eq. (15) is originally derived in [5] by showing that Eq. (12) is always implied by the sufficient RM schedulability condition that is proposed in [5].

4.2 Upper Bound on Work

We now derive an upper bound on the total amount of maximum work that can be completed by the jobs of the tasks in set $\{\tau_1, \tau_2, \ldots \tau_{k-1}\}$ over time interval $(0, t]$.

Over the interval $[0, t)$, exactly $\lfloor \frac{t}{T_i} \rfloor$ complete jobs of task τ_i are scheduled and the $(\lfloor \frac{t}{T_i} \rfloor + 1)^{th}$ job of τ_i may be scheduled for at most $min(t - \lfloor \frac{t}{T_i} \rfloor T_i, C_i)$ time units. The maximum amount of work completed by the jobs of task τ_i within any time interval $[0, t)$, which is at most $\lfloor \frac{t}{T_i} \rfloor C_i + min(t - \lfloor \frac{t}{T_i} \rfloor T_i, C_i)$, is upper bounded using the following Lemma 2 that is proved in [1, 10].

Lemma 2 (From [1, 10])**.** *The maximum amount of work* $\lfloor \frac{t}{T_i} \rfloor C_i + min(t - \lfloor \frac{t}{T_i} \rfloor T_i, C_i)$ *that can be completed by task* τ_i *over an interval* $[0, t)$ *is upper bounded by the following inequality in Eq.* (16)

$$\begin{aligned} &\lfloor \frac{t}{T_i} \rfloor C_i + min(t - \lfloor \frac{t}{T_i} \rfloor T_i, C_i) \\ &\qquad \leq \quad C_i + (t - C_i)\frac{C_i}{T_i} \end{aligned} \qquad (16)$$

By summing the Eq. (16) over $i = 1, 2, \ldots (k-1)$, the maximum amount of work completed by the jobs of the tasks in set $\{\tau_1, \tau_2, \ldots \tau_{k-1}\}$ over the time interval $(0, t]$ is upper bounded by the right-hand side of the following inequality in Eq. (17)

$$\begin{aligned} &\sum_{i=1}^{k-1} [\ \lfloor \frac{t}{T_i} \rfloor C_i + min(t - \lfloor \frac{t}{T_i} \rfloor T_i, C_i)\ ] \\ &\qquad \leq \sum_{i=1}^{k-1} [\ C_i + (t - C_i)\frac{C_i}{T_i}\ ] \end{aligned} \qquad (17)$$

Next we present our global RM schedulability analysis on uniform multiprocessors based on the inequalities given in Eq. (15) and Eq. (17).

5 Schedulability Analysis

In this section, we derive a sufficient schedulability condition for global RM scheduling on uniform multiprocessor platform. First, the maximum available time for executing the l^{th} job of task τ_k is calculated using the difference between the upper and lower bound on work completed by the RM scheduling over time intervals (0, lT_k] and (0, $(l-1)T_k$], respectively. Based on this difference, a sufficient schedulability condition is derived for task τ_k in Subsection 5.1 for $k = 1, 2 \ldots n$ (resulting in an iterative sufficient schedulability condition for the entire task set). Then in Subsection 5.2, the schedulability conditions of all the tasks are condensed into a simple (non-iterative) sufficient schedulability condition for uniform multiprocessors. Based on the simple sufficient schedulability condition derived in Subsection 5.2 for uniform multiprocessors, a corresponding simple suffient schedulability condition is derived for unit-capacity multiprocessors in Subsection 5.3.

5.1 Iterative Schedulability Condition for π

In this subsection, a sufficient RM schedulability condition of the task set Γ^n on uniform multiprocessor platform π is derived using the sufficient RM schedulability condition of each task τ_k, for $k = 1, 2, \ldots n$. The following Lemma 3 proves that all the jobs of task τ_k meet their deadlines using global RM scheduling on platform π if the conditions in Eq. (18) and Eq. (19) are satisfied (a sufficient schedulability condition for task τ_k).

Lemma 3. *All the jobs of* τ_k *meet their deadlines when* Γ^k *is scheduled on* π *using global RM scheduling if the two following conditions are satisfied:*

$$S(\pi) \geq U^k + \lambda(\pi) \cdot u^k_{max} \qquad (18)$$

and,

$$\left[\frac{S(\pi) - \mu(\pi) \cdot u_k}{(1 + r''_k)} + u_k + \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)} \right] \geq U^k \qquad (19)$$

Proof. Lets assume that all the $(l-1)$ jobs of τ_k have met their deadlines using RM. We will prove that the l^{th} job of τ_k also meets the deadline. The correctness of Lemma 3 will then follow by induction on $l \geq 1$.

Since each job of task τ_k is released at each period T_k, the l^{th} job arrives at time $(l-1)T_k$ and requires C_k units of execution time before its deadline lT_k . Remember that task set Γ^k is schedulable using algorithm OPT on uniform multiprocessor platform π_0 (Lemma 1). The platform π_0, that is shown to exist by construction in the discussion following Lemma 1, must satisfy $S(\pi_0) = U^k$ and $s_1(\pi_0) = u^k_{max}$. It follows from Eq. (18) that $S(\pi) \geq S(\pi_0) + \lambda(\pi) \cdot s_1(\pi_0)$ is true for the uniform multiprocessor platforms π and π_0. Therefore, the condition in Eq. (12) of Theorem 1 is true whenever the condition in Eq. (18) is true. Since the condition in Eq. (12) is true, using Eq. (15) we have

$$W(\text{RM}, \pi, \Gamma^k, (l-1)T_k\ ) \geq (l-1)T_k\ \sum_{i=1}^{k} u_i =$$

$$(l-1)T_k\ \sum_{i=1}^{k-1} u_i + (l-1)T_k u_k \qquad (20)$$

According to Eq. (20), the minimum amount of work completed by RM before the l^{th} job of τ_k arrives at time $(l-1)T_k$ is $(l-1)T_k\ \sum_{i=1}^{k-1} u_i + (l-1)T_k u_k$. Note that, prior to time instant $(l-1)T_k$, the amount of work generated for task τ_k is exactly $(l-1)T_k\ u_k$. Since we assume that all the $(l-1)$ jobs of task τ_k have met their deadlines (inductive hypothesis), the total work executed by RM for the higher-priority tasks $\tau_1\ , \tau_2\ , \ldots \tau_{k-1}$ is at least $(l-1)T_k\ \sum_{i=1}^{k-1} u_i$ prior to the time instant $(l-1)T_k$.

Lemma 2 ensures that the maximum amount of work that can be completed by all the higher-priority tasks $\tau_1\ , \tau_2\ , \ldots \tau_{k-1}$ over the interval $[0, lT_k)$ is bounded from above by $\sum_{i=1}^{k-1} [C_i\ + (lT_k\ - C_i\ )\frac{C_i}{T_i}]$. In the previous paragraph, we saw that at least $(l-1)T_k\ \sum_{i=1}^{k-1} u_i$ of this work is completed prior to time instant $(l-1)T_k$. Therefore, at most

$$\sum_{i=1}^{k-1} [(C_i\ + (lT_k\ - C_i\ )u_i) - ((l-1)T_k\ u_i)]$$
$$= \sum_{i=1}^{k-1} (C_i\ + (T_k\ - C_i\ )u_i)$$

amount of work remains to be executed after time instant $(l-1)T_k$ for all the higher priority tasks $\tau_1, \tau_2, \ldots \tau_{k-1}$.

The amount of processors capacity left unused by tasks $\tau_1\ , \tau_2\ , \ldots \tau_{k-1}$ during the interval $[(l-1)T_k\ , lT_k)$ on the uniform multiprocessor platform π is therefore at least

$$S(\pi) \cdot T_k\ - \sum_{i=1}^{k-1} (C_i\ + (T_k\ - C_i\ )u_i) \qquad (21)$$

Not all of this capacity is available to the l^{th} job of τ_k if several processors are available at the same time. In the worst-case, if all the $m(\pi)$ processors are available at the same time, the l^{th} job of τ_k can execute only on one processor. Since RM is greedy, if several processors are simultaneously available, then τ_k will execute upon the fastest available processor (Definition 3). If τ_k executes on the j^{th} fastest processor at some instant, then all other processors with speed $s_x(\pi)$ for $x < j$ are busy at this instant and does not contribute to the remaining capacity given in Eq. (21). Therefore, the fraction of total remaining capacity given in Eq. (21) that can be used for executing the l^{th} job of task τ_k at this instant is at least $\frac{s_j(\pi)}{[s_j(\pi)+s_{j+1}(\pi)+\ldots+s_{m(\pi)}(\pi)]}$. This reasoning is similar to that of used in [5]. According to Definition 2, we have $\frac{s_j(\pi)}{[s_j(\pi)+s_{j+1}(\pi)+\ldots+s_{m(\pi)}(\pi)]} \geq \frac{1}{\mu(\pi)}$. Consequently, the amount of processing capacity available to the l^{th} job of τ_k during the interval $[(l-1)T_k\ , lT_k)$ on the uniform multiprocessor platform π is at least

$$\frac{1}{\mu(\pi)} \left[S(\pi) \cdot T_k\ - \sum_{i=1}^{k-1} (C_i\ + (T_k\ - C_i\ )u_i) \right]$$

To guarantee that the l^{th} job of τ_k meets its deadline, we need this capacity to be at least as large as the execution time of τ_k; that is, we must have,

$$\frac{1}{\mu(\pi)} \left[S(\pi) \cdot T_k\ - \sum_{i=1}^{k-1} (C_i\ + (T_k\ - C_i\ )u_i) \right] \geq C_k$$

$$\equiv \frac{1}{\mu(\pi)} \left[S(\pi) \cdot T_k\ - \sum_{i=1}^{k-1} (C_i\ + \frac{T_k\ C_i}{Ti} - \frac{C_i^2}{T_i}) \right] \geq C_k$$

$$\equiv \frac{1}{\mu(\pi)} \left[S(\pi) - \sum_{i=1}^{k-1} (\frac{C_i}{T_k} + \frac{C_i}{T_i} - \frac{C_i^2}{T_i\ T_k}) \right] \geq \frac{C_k}{T_k}$$

$$\equiv \frac{1}{\mu(\pi)} \left[S(\pi) - \sum_{i=1}^{k-1} (\frac{C_i}{T_i} + \frac{C_i}{T_k}(1 - \frac{C_i}{T_i})) \right] \geq \frac{C_k}{T_k}$$

$\Leftarrow$ (Since $T_i \leq T_k$, using Eq. (4), we have $\frac{T_i}{r''_k} \leq T_k$)

$$\frac{1}{\mu(\pi)} \left[S(\pi) - \sum_{i=1}^{k-1} (\frac{C_i}{T_i} + \frac{r''_k \cdot C_i}{T_i}(1 - \frac{C_i}{T_i})) \right] \geq \frac{C_k}{T_k}$$

$$\equiv \frac{1}{\mu(\pi)} \left[S(\pi) - (1 + r''_k) \sum_{i=1}^{k-1} \frac{C_i}{T_i} + r''_k \sum_{i=1}^{k-1} \frac{C_i^2}{T_i^2} \right] \geq \frac{C_k}{T_k}$$

$$\equiv \frac{1}{\mu(\pi)} \left[S(\pi) - (1 + r''_k) \sum_{i=1}^{k-1} u_i + r''_k \sum_{i=1}^{k-1} u_i^2 \right] \geq u_k$$

$$\equiv S(\pi) - (1 + r''_k) \sum_{i=1}^{k-1} u_i + r''_k \sum_{i=1}^{k-1} u_i^2 \geq \mu(\pi) \cdot u_k$$

$$\equiv S(\pi) - (1 + r''_k)(U^k - u_k) + r''_k \sum_{i=1}^{k-1} u_i^2 \geq \mu(\pi) \cdot u_k$$

$$\equiv S(\pi) - \mu(\pi) \cdot u_k + (1 + r''_k)u_k + r''_k \sum_{i=1}^{k-1} u_i^2 \geq (1 + r''_k) \cdot U^k$$

$$\equiv \frac{S(\pi) - \mu(\pi) \cdot u_k}{(1 + r''_k)} + u_k + \frac{r''_k}{(1 + r''_k)} \sum_{i=1}^{k-1} u_i^2 \geq U^k$$

$\Leftarrow$ (Since $\text{Q}^k \leq \sum_{i=1}^{k-1} u_i^2$ using Eq. (7))

$$\left[\frac{S(\pi) - \mu(\pi) \cdot u_k}{(1 + r''_k)} + u_k + \frac{r''_k \cdot \text{Q}^k}{(1 + r''_k)} \right] \geq U^k$$

$\Leftarrow$ (*Premise in Eq.* (19) *of this Lemma*) □

Next the sufficient RM schedulability condition of the entire task set Γ^n is given in Theorem 2 based on the sufficient schedulability condition in Lemma 3.

Theorem 2. *An implicit deadline periodic task set Γ^n is schedulable using RM algorithm on a uniform multiprocessor platform π, if the following $(n+1)$ conditions are satisfied:*

$$S(\pi) \geq U^n + \lambda(\pi) \cdot u^n_{max} \tag{22}$$

and, for all k=1,2,... n,

$$\left[\frac{S(\pi) - \mu(\pi) \cdot u_k}{(1 + r''_k)} + u_k + \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)}\right] \geq U^k \tag{23}$$

Proof. If $S(\pi) \geq U^n + \lambda(\pi) \cdot u^n_{max}$, then $S(\pi) \geq U^k + \lambda(\pi) \cdot u^k_{max}$ because $U^n \geq U^k$ and $u^n_{max} \geq u^k_{max}$ for all $k = 1, 2, \ldots n$. According to Eq. (23), we also have that the condition $\left[\frac{S(\pi)-\mu(\pi)\cdot u_k}{(1+r''_k)} + u_k + \frac{r''_k \cdot \mathrm{Q}^k}{(1+r''_k)}\right] \geq U^k$ is true for all $k = 1, 2, \ldots n$. Therefore, the two conditions in Eq. (18) and Eq. (19) of Lemma 3 are true for all tasks τ_k for $k = 1, 2 \ldots, n$. Consequently, all the jobs of all the tasks in set Γ^n meet their deadlines using RM scheduling on a uniform multiprocessor platform π using Lemma 3. $\square$

Theorem 2 provides a sufficient RM schedulability condition that has to be checked iteratively for all the n tasks in task set Γ^n. Next we derive a simple sufficient RM schedulability condition for Γ^n.

5.2 Simple Schedulability Condition for π

In this subsection, we condense the set of $(n+1)$ conditions given in Theorem 2 to derive a *simple* sufficient schedulability condition in the sense that it is non-iterative and has to be checked only once for the entire task set Γ^n. We derive the simple sufficient schedulability condition in this subsection such that it implies all the $(n+1)$ conditions given in Eq. (22) and Eq. (23) of Theorem 2. Hence by the transitivity property of implication, if the simple condition is true, then the task set Γ^n is RM schedulable on platform π. The simple sufficient feasibility condition for uniform multiprocessor will be given in Theorem 3. Before Theorem 3 is presented, we need the following Lemma 4.

Lemma 4. *For a non-negative constant $c \leq 1$, the following inequality holds.*

$$U^n - c \cdot \mathrm{Q}^n \geq U^k - c \cdot \mathrm{Q}^k \tag{24}$$

Proof. We start with the following equality using Eq. (6)

$$U^n - c \cdot \mathrm{Q}^n = \sum_{i=1}^{k} u_i + \sum_{i=k+1}^{n} u_i - c \cdot [\sum_{i=1}^{n} u_i^2 - (u^n_{max})^2]$$

Equivalently,

$$U^n - c \cdot \mathrm{Q}^n = \sum_{i=1}^{k} u_i + \sum_{i=k+1}^{n} u_i$$
$$- c \cdot [\sum_{i=1}^{k} u_i^2 + \sum_{i=k+1}^{n} u_i^2 - (u^n_{max})^2] \tag{25}$$

Since $c \leq 1$, and $u_i \geq u_i^2$, we have $\sum_{i=k+1}^{n} u_i \geq c \cdot \sum_{i=k+1}^{n} u_i^2$. Therefore from Eq. (25), we have

$$U^n - c \cdot \mathrm{Q}^n \geq \sum_{i=1}^{k} u_i - c \cdot [\sum_{i=1}^{k} u_i^2 - (u^n_{max})^2]$$

Since $\mathrm{Q}^k = \sum_{i=1}^{k} u_i^2 - (u^k_{max})^2$ according to Eq. (6), we have

$$U^n - c \cdot \mathrm{Q}^n \geq \sum_{i=1}^{k} u_i - c \cdot [\mathrm{Q}^k + (u^k_{max})^2 - (u^n_{max})^2]$$

$$\equiv U^n - c \cdot \mathrm{Q}^n \geq \sum_{i=1}^{k} u_i - c \cdot \mathrm{Q}^k + c \cdot [(u^n_{max})^2 - (u^k_{max})^2]$$

Since $(u^n_{max})^2 \geq (u^k_{max})^2$, we have

$$U^n - c \cdot \mathrm{Q}^n \geq \sum_{i=1}^{k} u_i - c \cdot \mathrm{Q}^k$$
$$\equiv \; U^n - c \cdot \mathrm{Q}^n \geq U^k - c \cdot \mathrm{Q}^k$$

$\square$

We now derive the simple (non-iterative) sufficient condition for RM schedulability of a task set Γ^n on a uniform multiprocessor platform π in Theorem 3.

Theorem 3. *The task system Γ^n is RM schedulable on uniform multiprocessor platform π if*

$$\frac{S(\pi) - \mu(\pi) \cdot u^n_{max}}{(1 + r''_n)} + \delta + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_n)} \geq U^n \tag{26}$$

where

$$\delta = \begin{cases} u^n_{max} & \textit{if } \mu(\pi) > 1 + r''_n \\ u^n_{min} & \textit{otherwise} \end{cases}$$

Proof. We prove this theorem by showing that the condition in Eq. (26) implies the following $(n+1)$ conditions:

$$S(\pi) \geq U^n + \lambda(\pi) \cdot u^n_{max}$$

and, for all $k = 1, 2, \ldots n$,

$$\left[\frac{S(\pi) - \mu(\pi) \cdot u_k}{(1 + r''_k)} + u_k + \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)}\right] \geq U^k$$

given in Theorem 2. We prove these $(n+1)$ conditions of Eq. (22) and Eq. (23) below in Case (1) and Case (2), respectively, based on Eq. (26). Then the RM schedulability of Γ^n on platform π follows from Theorem 2.

Case (1): We start with the premise of this Theorem given in Eq. (26)

$$\frac{S(\pi) - \mu(\pi) \cdot u^n_{max}}{(1 + r''_n)} + \delta + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_n)} \geq U^n$$

Using Eq. (11), that is $\mu(\pi) = \lambda(\pi) + 1$, we have

$$\frac{S(\pi) - [\lambda(\pi) + 1] \cdot u^n_{max}}{(1 + r''_n)} + \delta + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_n)} \geq U^n$$

$$\equiv S(\pi) - [\lambda(\pi) + 1] \cdot u^n_{max} + \delta \cdot (1 + r''_n) + r'_n \cdot \mathrm{Q}^n \geq U^n \cdot (1 + r''_n)$$

$$\equiv S(\pi) - \lambda(\pi) \cdot u^n_{max} - u^n_{max} + \delta \geq U^n + r''_n(U^n - \delta - \frac{r'_n}{r''_n} \cdot \mathrm{Q}^n) \tag{27}$$

We have $\left(U^n - \delta - \frac{r'_n}{r''_n} \cdot \mathrm{Q}^n\right) \geq 0$ for $\delta = u^n_{max}$ or $\delta = u^n_{min}$ using Eq. (9) or Eq. (10), respectively. Therefore, from Eq. (27), we have

$$S(\pi) - \lambda(\pi) \cdot u^n_{max} - (u^n_{max} - \delta) \geq U^n$$

Since $\delta = u^n_{max}$ or $\delta = u^n_{min}$, we have $(u^n_{max} - \delta) \geq 0$. Therefore,

$$S(\pi) - \lambda(\pi) \cdot u^n_{max} \geq U^n$$

Since this equation is equivalent to $S(\pi) \geq U^n + \lambda(\pi) \cdot u^n_{max}$, the premise of this Theorem implies the first condition Eq. (22) of Theorem 2.

Case (2): We have (premise of this theorem)

$$\frac{S(\pi) - \mu(\pi) \cdot u^n_{max}}{(1 + r''_n)} + \delta + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_n)} \geq U^n$$

$\Rightarrow$ (Using Eq. (5) we have $r''_k \leq r''_n$)

$$\frac{S(\pi) - \mu(\pi) \cdot u^n_{max}}{(1 + r''_k)} + \delta + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_k)} \geq U^n$$

$\Rightarrow$ (Using Eq. (5) we have $r'_n \leq r''_k$, therefore)

$$\frac{S(\pi) - \mu(\pi) \cdot u^n_{max}}{(1 + r''_k)} + \delta + \frac{r''_k \cdot \mathrm{Q}^n}{(1 + r''_k)} \geq U^n$$

$$\equiv \frac{S(\pi) - \mu(\pi) \cdot u^n_{max}}{(1 + r''_k)} + \delta \geq U^n - \frac{r''_k \cdot \mathrm{Q}^n}{(1 + r''_k)} \tag{28}$$

Since $\frac{r''_k}{(1+r''_k)} \leq 1$, using Eq. (24) of Lemma 4 we have

$$\left[U^n - \frac{r''_k}{(1 + r''_k)} \cdot \mathrm{Q}^n\right] \geq \left[U^k - \frac{r''_k}{(1 + r''_k)} \cdot \mathrm{Q}^k\right] \tag{29}$$

Therefore, from Eq. (28) using Eq. (29) we have

$$\frac{S(\pi) - \mu(\pi) \cdot u^n_{max}}{(1 + r''_k)} + \delta \geq U^k - \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)}$$

$$\equiv \frac{S(\pi)}{(1 + r''_k)} + \delta - \frac{\mu(\pi) \cdot u^n_{max}}{(1 + r''_k)} \geq U^k - \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)} \tag{30}$$

The sufficient schedulability condition of this Theorem depends on the value of δ which can be u^n_{max} or u^n_{min} for $\mu(\pi) > 1 + r''_n$ or $\mu(\pi) \leq 1 + r''_n$, respectively. To that end, we have Subcase (2a) and Subcase (2b).

Subcase (2a): In this subcase, we consider $\mu(\pi) > 1 + r''_n$ and hence $\delta = u^n_{max}$ according to Eq. (26). From Eq. (30), using $\delta = u^n_{max}$ we have

$$\frac{S(\pi)}{(1 + r''_k)} + u^n_{max}[1 - \frac{\mu(\pi)}{(1 + r''_k)}] \geq U^k - \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)} \tag{31}$$

Since $r''_k < r''_n$ according to Eq. (5), this subcase condition $\mu(\pi) > 1 + r''_n$ implies $\mu(\pi) > 1 + r''_k$. This means that $[1 - \frac{\mu(\pi)}{(1+r''_k)}] < 0$. Also, using Eq. (1), we have $u^n_{max} \geq u^k_{max}$ for $k \leq n$. Therefore, Eq. (31) can be rewritten as:

$$\frac{S(\pi)}{(1 + r''_k)} + u^k_{max}[1 - \frac{\mu(\pi)}{(1 + r''_k)}] \geq U^k - \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)}$$

$\equiv$ (By rearranging)

$$\left[\frac{S(\pi) - \mu(\pi) \cdot u^k_{max}}{(1 + r''_k)} + u^k_{max} + \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)}\right] \geq U^k$$

Therefore for Subcase (2a), the second condition of Theorem 2 is implied by the premise of this Theorem 3.

Subcase (2b): In this subcase, we consider $\mu(\pi) \leq 1 + r''_n$ and hence $\delta = u^n_{min}$ according to Eq. (26). From Eq. (30), using $\delta = u^n_{min}$ we have

$$\frac{S(\pi)}{(1 + r''_k)} + u^n_{min} - \frac{\mu(\pi) \cdot u^n_{max}}{(1 + r''_k)} \geq U^k - \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)}$$

$\Rightarrow$(Using Eq. (1), we have $u^k_{max} \geq u^n_{min}$ for all $k \leq n$)

$$\frac{S(\pi)}{(1 + r''_k)} + u^k_{max} - \frac{\mu(\pi) \cdot u^n_{max}}{(1 + r''_k)} \geq U^k - \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)}$$

$\Rightarrow$(Using Eq. (1), we have $u^n_{max} \geq u^k_{max}$ for all $k \leq n$)

$$\frac{S(\pi)}{(1 + r''_k)} + u^k_{max} - \frac{\mu(\pi) \cdot u^k_{max}}{(1 + r''_k)} \geq U^k - \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)}$$

$\equiv$ (By rearranging)

$$\left[\frac{S(\pi) - \mu(\pi) \cdot u^k_{max}}{(1 + r''_k)} + u^k_{max} + \frac{r''_k \cdot \mathrm{Q}^k}{(1 + r''_k)}\right] \geq U^k$$

Therefore for Subcase (2b), the second condition of Theorem 2 is implied by the premise of this Theorem 3.

We have consequently shown that the first and second condition of Theorem 2 are implied by the premise of this Theorem in Case (1) and in the two subcases of Case (2), respectively. Since the (iterative) sufficient schedulability conditions of Theorem 2 are true whenever the premise of this Theorem is true, the task set Γ^n is RM schedulable on uniform multiprocessor π. □

5.3 Schedulability Condition for Unit-Capacity Multiprocessor

Using the simple sufficient condition of Theorem 3, we can derive a simple sufficient condition for RM schedulability of a task set on the subclass of uniform multiprocessor platforms, called *unit-capacity multiprocessors*.

Corollary 1. *A periodic task system Γ^n is RM schedulable on an unit-capacity multiprocessor platform π having m processors for $m \geq 2$, if*

$$\frac{m[1 - u^n_{max}]}{(1 + r''_n)} + u^n_{max} + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_n)} \geq U^n \tag{32}$$

Proof. According to Definition 1 and Definition 2, we have $S(\pi) = m(\pi) = m$ and $\mu(\pi) = m(\pi) = m$, respectively, for unit-capacity multiprocessor platform having m processors (each processor has speed one). Note that, for any task set Γ^n, we also have $r''_n \leq 1$ according to Eq. (5). Therefore $(1 + r''_n) \leq 2$, and we consider $m \geq 2$. Hence we have, $m \geq (1+ r''_n)$ which is equivalent to $\mu(\pi) \geq (1+ r''_n)$ for unit-capacity multiprocessor platform. Consequently, $\delta = u^n_{max}$ for the value of δ in the sufficient condition in Eq. (26) of Theorem 3. Substituting the value $S(\pi) = m$, $\mu(\pi) = m$ and $\delta = u^n_{max}$ in the sufficient condition of Eq. (26) of Theorem 3, we have

$$\frac{S(\pi) - \mu(\pi) \cdot u^n_{max}}{(1 + r''_n)} + u^n_{max} + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_n)} \geq U^n$$
$$\equiv \frac{m[1 - u^n_{max}]}{(1 + r''_n)} + u^n_{max} + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_n)} \geq U^n$$

□

6 Performance Comparison

In this section, we will now measure the improvement of our simple sufficient schedulability conditions for implicit-deadline tasks compared to the corresponding state-of-the-art schedulability conditions for global RM scheduling on uniform and unit-capacity multiprocessors.

Uniform Multiprocessors (RM Priority): Prior to this work, the best possible simple (non-iterative) sufficient RM schedulability condition in terms of worst-case system utilization on uniform multiprocessors is derived by Goossens and Baruah in [5] for implicit deadline task set. The schedulability condition from [5] is given in Theorem 4.

Theorem 4 (Goossens and Baruah [5])**.** *A periodic implicit deadline task system Γ^n is RM schedulable on an uniform multiprocessor platform π if*

$$\frac{S(\pi) - \mu(\pi) \cdot u^n_{max}}{2} \geq U^n \qquad (33)$$

The left-hand side of Eq. (33) is smaller than the left-hand side of Eq. (26). Therefore, it is evident that any task set satisfying Eq. (33) must also satisfy Eq. (26), and not necessarily vice versa. In other words, there are some task sets that satisfy Eq. (26) but do not satisfy Eq. (33). Hence, our derived simple sufficient RM schedulability condition given in Theorem 3 dominates the condition in Theorem 4 for uniform multiprocessors.

Unit-Capacity Multiprocessors (RM Priority): Prior to this work, the best possible simple (non-iterative) sufficient RM schedulability condition in terms of worst-case system utilization on unit-capacity multiprocessors is derived by Bertogna, Cirinei and Lipari [8] for implicit-deadline task sets. The schedulability condition from [8] is given in Theorem 5.

Theorem 5 (Bertogna, Cirinei, Lipari [8])**.** *An implicit-deadline periodic task system Γ^n is RM schedulable on a unit-capacity multiprocessor platform having m processors if*

$$\frac{m[1 - u^n_{max}]}{2} + u^n_{max} \geq U^n \qquad (34)$$

The left-hand side of Eq. (34) is smaller than the left-hand side of Eq. (32). Hence, our derived simple sufficient RM schedulability condition given in Corollary 1 dominates the condition in Theorem 5 for unit-capacity multiprocessors.

Other Fixed-Priority Scheduling: Prior to this work, the best simple (non-iterative) sufficient fixed-priority (other than RM priority) schedulability condition in terms of worst-case system utilization on unit-capacity multiprocessors is derived by Andersson using the `SM-US` scheduling algorithm [1] for implicit-deadline task sets. It is proved in [1] that the algorithm `SM-US` has the worst-case system utilization of $2m/(3 + \sqrt{5}) \approx$ 38.19% on m processors.

Our derived simple sufficient RM schedulability condition in Corollary 1 dominates the condition derived in [1] if $u^n_{max} \leq (\sqrt{5} - 1)/(3 + \sqrt{5})$. This is because if $u^n_{max} \leq (\sqrt{5} - 1)/(3 + \sqrt{5})$, then the left hand-side of Eq. (32) is at least equal to

$$\frac{m[1 - (\sqrt{5} - 1)/(3 + \sqrt{5})]}{(1 + r''_n)} + u^n_{max} + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_n)}$$
$$\equiv \frac{2}{(1 + r''_n)} \cdot \frac{2m}{(3 + \sqrt{5})} + u^n_{max} + \frac{r'_n \cdot \mathrm{Q}^n}{(1 + r''_n)} \geq \frac{2m}{(3 + \sqrt{5})}$$

In general, we can conclude that for other simple fixed-priority schedulability conditions that our simple RM schedulability conditions does not dominate, it is also true that those conditions do not dominate ours either.

Impact of Task-Set Parameters: A main contribution of this paper is the set of task-set parameters that are visible in the schedulability conditions in Theorem 3 and Corollary 1. By analyzing these conditions it can be observed that a positive effect on schedulability occurs in the following cases:

- The sum of squares of individual task utilization is high (*cf.* parameter Q^n in the numerator of the third term in the condition of Theorem 3 and Corollary 1).
- The maximum ratio between periods of any higher and any lower-priority task is small (*cf.* parameter r''_n in the denominator of the first and third terms in the condition of Theorem 3 and Corollary 1).
- The ratio between periods of the highest and the lowest priority task is large (*cf.* parameter r'_n in the numerator of the third term in the condition of Theorem 3 and Corollary 1).

A consequence of this is that, if system designers has the freedom to select the parameters of individual tasks (guided by these three strategies), a higher system utilization can be achieved compared to a less pro-active (less parameterized) design approach.

Controllability of Task-Set Parameters: The smaller the number of task parameters upon which a condensed task-set parameter depends, the higher is the *controllability* that task-set parameter provides to the system designer. The task-set parameter minimum period ratio r'_n depends only on the minimum and the maximum periods of a task set. The maximum period ratio r''_n depends on all possible pairs of periods of a task set. And the sum of squares of individual task utilization Q^n depends on the WCET and the periods of all the tasks. Therefore, r'_n provides the highest (and Q^n the lowest) degree of controllability among the three new task-set parameters introduced in the simple RM schedulability conditions in Theorem 3 and in Corollary 1 for uniform and unit-capacity multiprocessors, respectively.

Sensitivity Analysis: We conducted a number of experiments based on the different levels of *controllability* of the three new task set-parameters to measure their *impact* on system utilization for unit-capacity multiprocessors. In next subsection, we present our experimental results to estimate the amount of improvement of our simple schedulability test for unit-capacity processors in corollary 1 over the simple schedulability test for that of in Theorem 5. The conclusions from the experiments are that they not only corroborate the performance predictions made above but also show order-of-magnitude performance improvements (in terms of number of scheduled task sets) for carefully-selected (i.e. controlled) task-set parameters.

6.1 Experimental Evaluation

We have theoretically proved the dominance of our schedulability test in Corollary 1 (we call PJ test[2]) over the schedulability test in Theorem 5 (called BCL test) for unit-capacity-processors. To quantitatively estimate the degree of dominance of the PJ test over the BCL test, a series of experiments are conducted using randomly generated task sets.

6.1.1 Simulation Setup

We run a number of 36 experiments. Each experiment has five simulation parameters: m, $minU$, $maxU$, T_{max} and T_{min}. Each experiment is characterized by the value of m, the range $(minU, maxU]$ and the integer set $\{T_{min}, \ldots T_{max}\}$. The value of m denotes the number of processors we consider for an experiment. Each experiment is carried out for a number of randomly generated task sets. The utilization u_i of a randomly generated task τ_i of a task set is uniformly distributed within $(minU,\ maxU]$. The period T_i of the task τ_i is randomly selected from the set $\{T_{min}, \ldots T_{max}\}$. The worst case execution time C_i of task τ_i is then $U_i \times T_i$.

The number of processors we consider in our experiments are m = 2, 4, 6, and 8. This parameter m is used to measure the impact of increasing number of processors (scalability) on schedulability test. We consider three different utilization ranges (0, 0.5], (0.25, 0.75] and (0, 1] for $(minU,\ maxU]$. The utilization ranges (0, 0.5], (0.25, 0.75] and (0, 1] are used to experiment with *light*, *medium* and *mixed*[3] tasks, respectively. In order to see the impact of periods of a task set on the schedulability test, three different period sets $\{100, \ldots 1000\}$, $\{500, \ldots 1000\}$ and $\{750, \ldots 1000\}$ are considered for $\{T_{min}, \ldots T_{max}\}$. The *four* values of m, the *three* utilization ranges for $(minU, maxU]$ and the *three* period-sets for $\{T_{min}, \ldots T_{max}\}$ constitute our 36 different experiments.

For each experiment, a total of 100000 task sets are randomly generated such that each of the 100000 task sets pass the PJ test in corollary 1. The 100000 task sets are generated according to the following procedure:

1. Initially, we generate $m + 1$ tasks.
2. Then we verify if the generated task set passes the PJ test.
3. If the answer of the PJ test is positive, then it is counted as one of the 100000 task sets. Then, we verify the BCL test for this task set. And,
   (a) if the answer for the BCL test is positive, then this task set passes both the PJ and BCL tests.
   (b) if the answer for the BCL test is negative, then this task set only passes the PJ test and fails to pass the BCL test.
   (c) then by adding one new task, we extend this (old) task set to a new task set and return to Step 2.
4. If the answer of the PJ test is negative, then we discard this task set and go to Step 1.

For each experiment, the total number of task sets (of the 100000 task sets that pass PJ test) that pass the BCL test is counted in variable $\mathtt{BCL}_{count}$. The *dominance factor* of the PJ test, denoted by $D_{\mathtt{PJ}}$, is given using Eq. (35) for each experiment.

$$D_{\mathtt{PJ}} = \frac{100000 - \mathtt{BCL}_{count}}{100000} \times 100\% \qquad (35)$$

The higher the value of $D_{\mathtt{PJ}}$, the higher is the dominance of the PJ test over the BCL test. For example, if the dominance factor $D_{\mathtt{PJ}}$= 20%, then 20% (20000 task sets) of the 100000 task sets are not schedulable by the BCL test.

[2]The name of the test is given by concatenating the first characters of the authors' last names

[3]*mixed* task includes *heavy* tasks in addition to *light* and *medium* tasks

6.1.2 Simulation Result

The result of the 36 experiments are grouped into three categories (each category has 12 experiments) based on the three different sets for $\{T_{min}, \ldots T_{max}\}$. The result of the 12 experiments in each of the three categories are given in Table 1, Table 2, and Table 3 for $\{T_{min}, \ldots T_{max}\}$ equals to $[100, \ldots 1000]$, $[500, \ldots 1000]$ and $[750, \ldots 1000]$, respectively. Each of the shaded cells in Table 1, Table 2, and Table 3 represents the value of $D_{\mathtt{PJ}}$ for simulation parameters — number of processors and utilization range — given in the corresponding first column and second row of each table, respectively.

Table 1. Value of $D_{\mathtt{PJ}}$ for the 12 experiments using $\{T_{min} \ldots T_{max}\}$=$\{$100 $\ldots$ 1000$\}$

	$(minU, maxU]$		
	(0, 1]	(0, 0.5]	(0.25, 0.75]
$m = 2$	21.42 %	15.56 %	67.14 %
$m = 4$	16.94 %	11.12 %	63.48 %
$m = 6$	16.74 %	10.46 %	63.50 %
$m = 8$	16.20 %	10.30 %	63.32 %

Table 2. Value of $D_{\mathtt{PJ}}$ for the 12 experiments using $\{T_{min} \ldots T_{max}\}$=$\{$500 $\ldots$ 1000$\}$

	$(minU, maxU]$		
	(0, 1]	(0, 0.5]	(0.25, 0.75]
$m = 2$	20.18 %	16.92 %	63.74 %
$m = 4$	23.80 %	17.08 %	73.80 %
$m = 6$	29.56 %	21.28 %	81.24 %
$m = 8$	35.3 %	24.52 %	87.46 %

Table 3. Value of $D_{\mathtt{PJ}}$ for the 12 experiments using $\{T_{min} \ldots T_{max}\}$=$\{$750 $\ldots$ 1000$\}$

	$(minU, maxU]$		
	(0, 1]	(0, 0.5]	(0.25, 0.75]
$m = 2$	21.06 %	18.08 %	63.92 %
$m = 4$	27.28 %	22.08 %	79.28 %
$m = 6$	37.02 %	27.98 %	88.26 %
$m = 8$	45.48 %	31.96 %	93.46 %

6.1.3 Result Analysis

Observe that the `PJ` test in Corollary 1 provides better RM schedulability (higher system utilization) if minimum period ratio r'_n is large, maximum period ratio r''_n is small and/or sum of square of utilization of individual task utilization Q^n is large.

The impact of utilization ranges $(minU, maxU]$ on the dominance of `PJ` test over `BCL` test is significant. The value of dominance, that is $D_{\mathtt{PJ}}$, for the `PJ` test over the `BCL` test using medium tasks (i.e. u_i is within [0.25, 0.75]) is significantly higher (see the forth columns of each Table) than that of using light and mixed tasks. The `BCL` test fails to schedule more than 63% task sets of the 100000 task sets that are schedulable by the `PJ` test. This is because of the use of the task-set parameter sum of square (i.e. Q^n) in the `PJ` test. Each task τ_i adds u_i^2 (except ${u_{max}^n}^2$) to the calculation of Q^naccording to Eq. (6). Since all tasks are medium, the value of $u_i^2 > 0.25^2 = 0.0625$. The higher the value of u_i, the higher is the value of $Q^n$and the higher is the system utilization. Consequently, the value of $D_{\mathtt{PJ}}$ is comparatively much higher using medium tasks than that of using light or mix tasks. *In summary, `PJ` tests significantly outperforms `BCL` tests for medium tasks.*

The relationship of the periods of a task set has varying impacts on the dominance factor of the `PJ` test over the `BCL` test. Observe that, according to Eq. (2) the minimum period ratio r'_n of a task set is the ratio between the minimum and maximum periods (i.e. one pair of periods) of a task set. As a result when the periods of the tasks of a task set are randomly selected from a set of denser integer values, then the value of r'_n can be expected to be higher than that of when the periods are selected from a set of dispersed integer values. The values of $D_{\mathtt{PJ}}$ in most of the shaded cells in Table 3 are larger than that of the corresponding cells in either Table 1 or Table 2 since the task periods in Table 3 are selected from a denser set $\{750 \ldots 1000\}$ than that of in either Table 1 or Table 2.

The value of maximum period ratio r''_n depends on the number of tasks, number of processors and the set $\{T_{min} \ldots T_{max}\}$ that are used for selecting the periods of the randomly generated tasks of a task set. Recall that, according to Eq. (3), the maximum period ratio r''_n of a task set depends on all different pairs of periods of a task set. For a set of n tasks, there are $\binom{n}{2} = \frac{n!}{(n-2)!\cdot 2!}$ different pairs of periods. The higher is the number of tasks in a task set, the higher is the number of different pairs of periods. The higher is the number of different pairs of periods, the higher is the possibility that the values of two periods are closer. And the closer the two periods are, the higher is the value of maximum period ratio r''_n (according to Eq. (3)). Also note that the way tasks are generated for a task set implies that the higher is the number of processors, the higher is the number of tasks and hence the higher is the probability that the value of r''_n is larger. And according to Eq. (6), the value of Q^n increases as more tasks are included in a task set. Therefore, as we increase the number of processors, both the values of r''_n and Q^n increases.

Each of the 12 experimental results in Table 1 are derived based on tasks' periods that are selected from a set of relatively dispersed integer values. Therefore, we expect the value of minimum period period r'_n of each task set to be relatively small. As we increase the number of processors even though the value of Q^n increases (positive impact factor on schedulability), the smaller value of r'_n and larger value of r''_n (negative impact factors on schedulability) result in decreasing value of dominance of the PJ test over BCL test for all the three utilization ranges (see the decreasing trend in each column of Table 1). The negative impact factors (i.e. low r'_n and high r''_n) can not be offset by the positive impact factor (i.e. high Q^n) for the simulation parameters used to generate the values given in Table 1. Therefore, as we increase the number of processors (i.e. increase in number of tasks in a task set), the value of $\mathrm{D}_{\mathtt{PJ}}$ decreases for all types (light, medium and mixed) of tasks.

However, the dominance factor increases as we increase the number of processors for each of the utilization ranges as shown in Table 2 and Table 3. This is because even though the value of r''_n increases (negative impact factor on schedulability) as we increase the number of processors, the value of Q^n increases and the value of r'_n is relatively larger for using a relatively denser set for periods. The negative impact of the higher value of r''_n is offset by the positive impact of the higher value of r'_n and Q^n. Consequently, as we increase the number of processors, the value of $\mathrm{D}_{\mathtt{PJ}}$ increases for all three utilization ranges for Table 2 and Table 3. *In summary, if the ratio of the minimum and maximum period of a task set are closer (value of r'_n is larger), then the PJ test scales well with increasing number of processors for any kind (light, medium and mix)* of task sets.

7 Conclusion

In this paper we have proposed schedulability conditions for global rate-monotonic scheduling of implicit-deadline tasks on multiprocessors. The prominent feature of our conditions is that they introduce a set of new task-set parameters that are key to achieving good schedulability performance in terms of worst-case system utilization. The two proposed simple RM schedulability conditions for uniform and unit-capacity processors are shown to dominate any other known simple RM schedulability condition. For other simple fixed-priority schedulability conditions that our conditions does not dominate, it is shown that those conditions do not dominate our schedulability condition either.

References

[1] B. Andersson. Global Static-Priority Preemptive Multiprocessor Scheduling with Utilization Bound 38%. *in Proc. of Prin. of Dist. Syst.*, pages 73–88, 2008.

[2] B. Andersson, S. Baruah, and J. Jonsson. Static-Priority Scheduling on Multiprocessors. *in Proc. of RTSS*, pages 193–202, 2001.

[3] B. Andersson and J. Jonsson. The utilization bounds of partitioned and pfair static-priority scheduling on multiprocessors are 50%. *in Proc. of ECRTS*, pages 33–40, 2003.

[4] T. P. Baker. An Analysis of Fixed-Priority Schedulability on a Multiprocessor. *Real-Time Systems*, 32(1-2):49–71, 2006.

[5] S. Baruah and J. Goossens. Rate-Monotonic Scheduling on Uniform Multiprocessors. *IEEE Trans. on Comput.*, 52(7):966–970, 2003.

[6] S. Baruah and J. Goossens. Deadline Monotonic Scheduling on Uniform Multiprocessors. *in Proc. of Prin. of Dist. Syst.*, pages 89–104, 2008.

[7] O. Beaumont, L. Arnaud, and Y. Robert. Static Scheduling Strategies for Heterogeneous Systems. *Computing and Informatics*, 21:413–430, 2002.

[8] M. Bertogna, M. Cirinei, and G. Lipari. New Schedulability Tests for Real-Time Task Sets Scheduled by Deadline Monotonic on Multiprocessors . *in Proc. of Prin. of Dist. Syst.*, pages 306–321, 2005.

[9] M. Bertogna, M. Cirinei, and G. Lipari. Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms. *IEEE Trans. on Parallel and Dist. Syst.*, 20(4):553–566, 2009.

[10] E. Bini, T. H. C. Nguyen, P. Richard, and S. Baruah. A Response-Time Bound in Fixed-Priority Scheduling with Arbitrary Deadlines. *IEEE Trans. Comput.*, 58(2):279–286, 2009.

[11] S. Chaudhry, R. Cypher, M. Ekman, M. Karlsson, A. Landin, S. Yip, H. Zeffer, and M. Tremblay. Rock: A High-Performance Sparc CMT Processor. *IEEE Micro*, 29(2):6–16, 2009.

[12] R. I. Davis and A. Burns. Priority Assignment for Global Fixed Priority Pre-Emptive Scheduling in Multiprocessor Real-Time Systems. In *Proc. of RTSS*, pages 398–409, 2009.

[13] S. K. Dhall and C. L. Liu. On a Real-Time Scheduling Problem. *Operations Research*, 26(1):127–140, 1978.

[14] S. Funk, J. Goossens, and S. Baruah. On-Line Scheduling on Uniform Multiprocessors. *in Proc. of RTSS*, pages 183–192, 2001.

[15] T. Gonzalez, O. H. Ibarra, and S. Sahni. Bounds for LPT Schedules on Uniform Processors. *SIAM Journal on Computing*, 6(1):155–166, 1977.

[16] J. Goossens, S. Funk, and S. Baruah. Priority-Driven Scheduling of Periodic Task Systems on Multiprocessors. *Real-Time Systems.*, 25(2-3):187–205, 2003.

[17] H. Jin and P. Tan. A Novel Dynamic Allocation and Scheduling Scheme with CPNA and FCF Algorithms in Distributed Real-time Systems. In *Proc. of ICPADS*, pages 550–556, 2005.

[18] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

[19] J. M. López, M. García, J. L. Díaz, and D. F. García. Utilization Bounds for Multiprocessor Rate-Monotonic Scheduling. *Real-Time Systems*, 24(1):5–28, 2003.

[20] L. Lundberg. Analyzing Fixed-Priority Global Multiprocessor Scheduling. *in Proc. of RTAS*, pages 145–153, 2002.

[21] D.-I. Oh and T. P. Baker. Utilization Bounds for N-Processor Rate Monotone Scheduling with Static Processor Assignment. *Real-Time Systems*, 15(2):183–192, 1998.

[22] L. Sha, J. P. Lehoczky, and R. Rajkumar. Solutions for Some Practical Problems in Prioritized Preemptive Scheduling. In *Proc. of RTSS*, pages 181–191, 1986.