Efficient On-Line Schedulability Tests for Real-Time Systems

Tei-Wei Kuo, Yu-Hua Liu†, Li-Pin Chang, and Kwei-Jay Lin‡
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan 106, ROC
Email: {ktw,d6526009}@csie.ntu.edu.tw, Fax: +886-223628167
†Department of Computer Science and Information Engineering
National Chung Cheng University, Chiayi, Taiwan 621, ROC
‡Department of Electrical and Computer Engineering
University of California, Irvine, Irvine, CA 92697, USA

Abstract

Many computer systems, such as those for open system environments or multimedia
services, need an efficient schedulability test for on-line admission control of new jobs.
Although various polynomial time schedulability tests have been proposed, they often fail
to decide the schedulability of the system precisely when the system is heavily loaded. On
the other hand, most precise schedulability tests proposed to date have a high complexity
and may not be suitable for on-line tests. In this paper, we present new efficient on-line
schedulability tests for both the periodic process model [22] and the multiframe process
model [26, 27] in uni-processor environments. The schedulability tests are shown to be more
precise and efficient than any existing polynomial-time schedulability tests. Moreover, the
tests can be done incrementally as each new task arrives at the system. Our proposed tests
can also be used for the multiframe model where a task may have different computation
times in different periods. We show the performance of the proposed schedulability tests in
several simulation experiments.

Keywords: Real-Time Systems, Schedulability Test, Division Graph, Reduced Set, Multiframe
Process, Open System Environment, MPEG Streams, Time Reservation.

*Supported in part by a research grant from the ROC National Science Council under Grants NSC87-2213-E-
194-002 and by USA NSF CCR-9901697
†This paper is an extended version of a conference paper in the IEEE Real-Time Technology and Applications
1 Introduction

Computer servers with real-time capabilities are in great demands. Many of our daily activities, both personal and business, must make use of some computers in some contiguous interval (e.g., an hour or so). These activities must be done in an interactive and real-time manner (e.g., on-line entertainment, video conference, distance learning, etc.) The servers therefore must be able to handle a dynamic workload and yet provide a predictable performance. One recent proposal to build real-time servers for complex environments is the model of open systems. In an open real-time system environment [5, 6, 15], various applications may run concurrently using different application schedulers. Due to the dynamic nature of the system workload, the admission test must admit or reject applications (or threads) dynamically, precisely, and efficiently. A new application (or thread) can be admitted only if the available resources can serve both the new application (or thread) and all existing applications (or threads) without violating any individual service requirement.

The schedulability test for priority-based task set has been extensively studied [2, 4, 9, 13, 19, 22, 26, 27, 33, 36]. Some of them are known to be precise [4, 19, 33, 36]. However, precise tests usually have high time complexities and may not be adequate for on-line admission control if the system has a large number of tasks or a dynamic workload. Efficient on-line schedulability tests are needed for many computer systems with performance guarantee, such as those consider open system environments [5, 6, 15] or time reservation [7, 16, 28, 29, 30, 31], where real-time and non-real-time applications may be admitted in a dynamic fashion. Efficient on-line schedulability tests are also useful to various service-critical systems, such as tele-medicine systems, tele-conferencing systems, and multimedia services that have Quality-of-Service (QoS) requirements [11, 14, 37].

In addition to their complexity, many precise tests for priority-driven systems require iterative calculations of the task response time in the presence of higher priority tasks. It is therefore difficult to develop an intuitive understanding on how a timing parameter change in one task may affect the rest of the system schedulability. Closed-form (but imprecise) tests, on the other hands, give a more intuitive reasoning on the effect of changing certain timing parameters on the system performance. On the other hand, efficient schedulability tests provide on-line polynomial-time tests. For example, harmonic-period based schedulability tests [8, 9, 10, 13] utilize the jitterless behavior among tasks with harmonic periods and use it to increase the accuracy of the schedulability. Since many real-time tasks can be designed to execute in any period size, selecting a task period wisely to make it harmonic to other periods will have a big payoff in the system schedulability. Similarly, many multimedia tasks or control tasks have only a few base execution frequencies. By analyzing the system schedulability based only on these basic (or root) period
values, we show in this paper that the system performance can be more accurately predicted.

In this paper, we present new efficient on-line schedulability tests for both the periodic process model [22] and the multiframe process model [26, 27] in uniprocessor environments. The major contributions of this work are as follows:

1. We exploit the harmonic relationship of task periods to provide a more precise schedulability test for the periodic process model [22]. The proposed schedulability test not only is much more precise than existing schedulability tests [2, 9, 10, 13, 22] but also has an efficient implementation.

2. Our proposed methodology can also be used for the multiframe process model [26, 27] when a task may have a different execution time requirement in different periods.

3. An incremental schedulability test algorithm has been proposed for on-line admission control of newly arrived tasks. The incremental algorithm does not need to recompute the response time of all tasks as in some precise schedulability tests [4, 19, 33, 36].

4. Our tests are especially effective for systems with heavy loads. As will be shown later in this paper, our proposed schedulability tests are shown to be much more precise than any existing polynomial-time schedulability tests [2, 9, 10, 13, 22, 26] when the system is heavily loaded.

We must point out that the techniques proposed in this paper is orthogonal to the past work in the overhead modeling and complicated scheduling algorithms, such as pre-period deadlines, sporadic servers, process synchronization. The goal of this paper is to provide a theoretical base for more efficient schedulability tests. The concept developed in this paper could be integrated with the work in the Rate Monotonic Analysis [19, 33] regarding the overhead handling of context switches and sporadic-server-based analysis [43] and the work in process synchronization regarding blocking time analysis, e.g., [34].

The rest of this paper is organized as follows. Section 2 provides an overview of well-known schedulability tests and compare their pros and cons. Section 3 presents our precise polynomial-time schedulability test for the Liu&Layland model [22]. We provide a theoretical foundation for efficient schedulability tests by exploring the harmonic relationship of task periods. An efficient incremental algorithm is presented for on-line admission control. Section 4 extends the proposed schedulability test for the multiframe process model [26, 27]. The simulation results for the proposed schedulability tests for the Liu&Layland process model and the the multiframe process model are shown in Section 5. Our tests are especially effective for systems with heavy loads. Our proposed schedulability tests are shown to be much more precise than any existing
Table 1: \(p_i\) and \(c_i\) denote the period \(p_i\) and the computation requirement of process \(\tau_i\), respectively. \(U_i\) is the accumulative CPU utilization factor of processes up to \(\tau_i\).

2 Overview

The real-time resource allocation problem has been an active research topic in recent decades. Liu and Layland [22] first introduced the concept of achievable utilization factor to provide a quick test for deciding the schedulability of a set of independent periodic processes with fixed execution time and period. In [8], Han and Lin define the temporal distance constraints among processes and investigate systems with harmonic task periods. The concept of harmonic task periods has also been investigated by Kuo and Mok [13] in their study of adaptive system workloads. More recently, Han and Tyan [9, 10] proposed new and more precise polynomial-time schedulability tests for the Liu&Layland (independent) process model [22], which were shown to be better than the results proposed in [2, 22]. The proposed schedulability test is based on the comparison with a system consisting only of tasks with harmonic periods.

Under the Liu&Layland (independent) process model [22], the resource requirements of processes are fixed in every period, which can be modeled by two parameters \((c_i, p_i)\). In this model, a task \(\tau_i\) must be serviced for \(c_i\) time units per \(p_i\) time interval. The utilization factor [22] of a system is defined by

\[
U = \sum_{i=1}^{n} \frac{c_i}{p_i}
\]

where \(c_i\) and \(p_i\) are the computation requirement and period of the process \(\tau_i\), respectively. \(\frac{c_i}{p_i}\) is called the utilization factor of the process \(\tau_i\). For the simplicity of presentation, we will use terminologies “task” and “process” interchangeably for the rest of the paper.

The on-line admission control of a new task \(\tau_i = (c_i, p_i)\) can be done in terms of the concept of the achievable utilization factor [22]. The schedulability of a set of \(n\) processes is guaranteed if the utilization factor of the process set is no more than the achievable utilization factor \(U(n) = n(2^{1/n} - 1)\) of the Rate Monotonic Scheduling (RMS) algorithm [22], which is an optimal fixed-
priority scheduling algorithm. For example, let $T$ be a set of four processes with period $p_i$ and computation requirement $c_i$, summarized in Table 1. Note that $U_i$ denotes the accumulative CPU utilization factor of processes $\tau_1$, $\tau_2$, ..., and $\tau_i$. The arrival of a new process $\tau$ with period 20 and computation requirement 3 would fail the admission control of the system because the utilization factor of the system becomes 95% and is far over the achievable utilization factor $U(5) = 5(2^{1/5} - 1) = 74.3\%$ reported in [22]. As astute readers may point out, the utilization factor of $T$ already fails the achievable utilization factor $U(4) = 4(2^{1/4} - 1) = 75.7\%$ reported in [22].

Although Liu and Layland [22] did provide an $O(n)$ polynomial-time schedulability test for on-line admission control, their schedulability test fails to identify many schedulable process sets, when the system is heavily loaded with a number of processes. The schedulability test proposed by Kuo and Mok [13] explored the harmonic relationship of session periods and was proved to be much better than the results of Liu and Layland [22]. They show that a process set with periods selected from $K$ fundamental frequencies is schedulable if the utilization factor of the process set is no more than $U(K) = K(2^{1/K} - 1)$. For example, there are two fundamental frequencies in process set $T$ in Table 1, and the schedulability of process set $T$ is guaranteed under the schedulability test proposed by Kuo and Mok [13] because the utilization factor of $T$ is no larger than $U(2) = 2(2^{1/2} - 1) = 82.8\%$. However, the arrival of new process $\tau$ (in the previous paragraph) still fails the schedulability test proposed by Kuo and Mok [13] because the utilization factor of the system becomes 95% and is still over $U(2) = 2(2^{1/2} - 1) = 82.8\%$. We should point out the schedulability test has a time complexity $O(n^{5/2})$ and cannot be easily implemented in an incremental way. The schedulability test also quickly deteriorates when the number of fundamental frequencies increases, or when the system is heavily loaded.

Recently, Han and Tyan [9] proposed a more precise polynomial-time schedulability test, which were shown to be better than the results proposed in [2, 22]. The proposed schedulability test is based on the transformation of process periods into a special pattern, i.e., a single fundamental

<table>
<thead>
<tr>
<th>$\tau_i$</th>
<th>$\tau_1$</th>
<th>$\tau_2$</th>
<th>$\tau_3$</th>
<th>$\tau_4$</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>original $p_i$</td>
<td>3</td>
<td>5</td>
<td>15</td>
<td>60</td>
<td>20</td>
</tr>
<tr>
<td>original $c_i$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>original $U_i$</td>
<td>0.3333</td>
<td>0.5333</td>
<td>0.6666</td>
<td>0.8</td>
<td>0.95</td>
</tr>
<tr>
<td>candidate 1 $p_i$ ($r = 1.875$)</td>
<td>1.875</td>
<td>3.75</td>
<td>15</td>
<td>60</td>
<td>15</td>
</tr>
<tr>
<td>candidate 1 $U_i$</td>
<td>0.5333</td>
<td>0.8</td>
<td>0.9333</td>
<td>1.0666</td>
<td>1.2666</td>
</tr>
<tr>
<td>candidate 2 $p_i$ ($r = 2.5$)</td>
<td>2.5</td>
<td>5</td>
<td>10</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>candidate 2 $U_i$</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>1.0</td>
<td>1.15</td>
</tr>
<tr>
<td>candidate 3 $p_i$ ($r = 3$)</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>48</td>
<td>12</td>
</tr>
<tr>
<td>candidate 3 $U_i$</td>
<td>0.3333</td>
<td>0.6666</td>
<td>0.8333</td>
<td>1.0</td>
<td>1.25</td>
</tr>
</tbody>
</table>

Table 2: The period $p_i$ and computation requirement $c_i$ of each process before and after transformation [9].
frequency. For example, \( p'_i = r \cdot 2^{\lfloor \log_2(p_i/r) \rfloor} \), where \( p_i \) and \( p'_i \) are the original and new periods of each process \( \tau_i \), respectively. The selection of \( r \) must minimize the increase of the utilization factor after transformation. If the transformed process set is schedulable, then the original process set is schedulable. Since the periods of the transformed process set form a single fundamental frequency, the original process set is schedulable if the utilization factor of the transformed process set is no more than 100%. For example, process set \( T \) described in the previous paragraph has three candidates for \( r \), where \( r \in \left\{ \frac{p_i}{2^{\lfloor \log_2(p_i/p_1) \rfloor}} \right\} \). Table 2 show the transformed period \( p'_i \) of each process when a different \( r \) is selected. Obviously, \( r = 2.5 \) and 3 result in the minimum increase of the utilization factor, and the original process set is schedulable because the utilization factor of the transformed process set is no more than 100%. However, the arrival of new process \( \tau \) (in the previous paragraph) still fails the schedulability test proposed by Han and Tyan [9] because, regardless of which \( r \) is selected, the utilization factor of the transformed process set will be far more than 100%. In fact, the utilization factors of the transformed process set will become 1.2666, 1.15, and 1.25 when \( r = 1.875, 2.5 \) or 3 are selected, respectively. As astute readers may point out, the transformation proposed by Han and Tyan [9] usually results in the significant increase of processor utilization and damage the preciseness of their proposed schedulability test, especially at heavy system load. Furthermore, the schedulability test is quite complicated and has a time complexity \( O(n^2) \) for a system of \( n \) tasks and can not be implemented with an incremental algorithm because of the recalculation of the special base and, most importantly, the procedure in minimizing the increase of utilization factor.

In the following sections, we shall propose a polynomial-time schedulability test which is more precise and efficient than existing polynomial-time tests [2, 9, 13, 22]. We will show later that the proposed schedulability test significantly outperforms any other schedulability test [9, 13, 22], especially when the system is heavily loaded. The proposed schedulability test therefore meets the needs of many computer systems which can be moderately or heavily loaded and require efficient on-line admission control. We will then extend our results to the multiframe process model [26, 27] when a task may have a different execution time requirement in different periods.

3 Efficient Schedulability Test for the Liu&Layland Model

3.1 Schedulability Test Based on the Liu&Layland Model

The purpose of this section is to propose more precise efficient schedulability tests. We show how the harmonic relationship could be used to improve the precision and the efficiency of schedulability tests. A basic schedulability test and its incremental implementation are then
presented. We shall show later in the performance evaluation that the proposed schedulability test outperforms any other existing schedulability tests [9, 13, 22].

3.1.1 Definitions

Before proceeding with the following discussion and proofs, some definitions and concepts from [13, 22] are in order.

Definition 1 Division Graph [13]:

Given a set of real numbers \( P = \{e_1, e_2, \ldots, e_n\} \), we construct an irreflexive, asymmetric, transitive, acyclic directed graph \( G = (V, E) \) such that

1. Every vertex \( \in V \) has a corresponding \( e_i \in P \), i.e., \(|P| = |V|\).
2. There is a directed edge \((e_i, e_j) \in E \) if \( e_j | e_i \), where \( p_j | p_i \) means \( p_i \) is an exact multiple of \( p_j \).
3. No arc \((e_i, e_j) \) exists if \( e_j \not{|} e_i \).

A division graph represents the divisibility relation among a set of real numbers. Given a process set, the corresponding division graph of the periods of its processes can be obtained in a straightforward way.

Definition 2 Offspring Set:

Let \( T = \{\tau_1, \ldots, \tau_i, \ldots, \tau_n\} \) be a set of periodic processes. An offspring set of process \( \tau_i \) in \( T \) is a subset of \( T \) in which the period of each process divides the period of \( \tau_i \).

We shall show later that processes in an offspring set would be merged together to provide a better schedulability test. There is no need to let an offspring set being the largest such set because we can have more flexibility in merging processes in an offspring set of whatever size.

Definition 3 Root:

Let \( T = \{\tau_1, \ldots, \tau_i, \ldots, \tau_n\} \) be a set of periodic processes. Process \( \tau_i \) is a root in \( T \) if there does not exist any process period in \( T \) which is larger than and can be divided by the period of \( \tau_i \).

By the definition of roots, there can be more than one root process in a given process set. Each root process may be associated with more than one offspring set, and the offspring sets of different (root) processes may have intersection.
Definition 4 Reduced Set and RSR:

Suppose that $\tau$ is a periodic process with period $p$ and computation requirement $c$. A set of $n$ periodic processes, $\{\tau_1, \cdots, \tau_i, \cdots, \tau_n\}$ where process $\tau_i$ has period $p_i$ and computation requirement $c_i$, is a reduced set of $\tau$ if $\sum_{i=1}^{n} \frac{c_i}{p_i} = \frac{c}{p}, \forall p_i, p_i | p$, and $p = \max_{i=1}^{n} p_i$, where $p_j | p_i$ means $p_i$ is an exact multiple of $p_j$. $\tau$ is called the RS-representative, or RSR, of the reduced set.

Example 1 Reduced sets, roots, and offspring sets

![Division Graph of Example 1]

Figure 1: The division graph of Example 1

Let $T = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$ be a set of five processes, and the periods of $\tau_1, \tau_2, \tau_3, \tau_4$, and $\tau_5$ are 3, 5, 15, 20, and 60, respectively. The division graph of the set of the periods of processes in $T$ is shown in Figure 1. There is a single root, i.e., 60, in the division graph of $T$, and there are several offspring sets of $\tau_5$, e.g., $\{\tau_1, \tau_2, \tau_4\}$ and $\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$. There are also offspring sets for other processes in $T$. For example, $\{\tau_1, \tau_2\}$ is an offspring set of $\tau_3$. A process $\tau$ is a RSR of a set $\{\tau_1, \tau_2, \tau_3\}$ if the period of $\tau$ is 15, and the utilization factor of $\tau$ is equal to the sum of the utilization factors of $\tau_1, \tau_2$, and $\tau_3$. $\{\tau_1, \tau_2, \tau_3\}$ is a reduced set of $\tau$. □

3.1.2 Schedulability Theorems

We are interested in uniprocessor, fixed-priority-driven scheduling of independent hard real-time processes. The deadline of each request of a process is the next request time. Let $T = \{\tau_1, \cdots, \tau_i, \cdots, \tau_n\}$ be a set of periodic processes, where $c_i$ and $p_i$ are the computation requirement and period of process $\tau_i$, respectively. Assume $p_i$‘s are sorted in increasing order. Since the Rate Monotonic Scheduling (RMS) algorithm is an optimal fixed-priority scheduling algorithm, RMS is adopted in this paper. Suppose that the periods of processes in $T$ are all different. Note that this constraint can be relaxed easily by merging processes with the same period into one process and, then, solving the schedulability problems (by applying the following theorems).

Let $T_i = \{\tau_1, \cdots, \tau_i\}$ be a subset of $T$ with processes whose periods are less than or equal to $p_i$. When we say that a process set $TS$ is schedulable, it means that every process in $TS$ is
schedulable at its critical instant (i.e., within its critical interval). We can prove the following theorems based on the concept of root and reduced set:

**Theorem 1** [19, 33] Process \( \tau_i \) in a set of periodic processes scheduled by a fixed-priority-driven preemptive scheduling algorithm will always meet its deadline for all process phases iff there exists a pair \((k, m) \in R_i \) such that

\[
\sum_{j<i} (c_j \left\lceil \frac{mp_j}{p_j} \right\rceil) + c_i \leq mp_k,
\]

where \( R_i = \{(k, m) | 1 \leq k \leq i, m = 1, 2, \cdots, \left\lfloor \frac{p_i}{p_k} \right\rfloor \} \).

Each pair \((k, m)\) represents a scheduling time point \( mp_k \) to test the schedulability of process \( \tau_i \). \( R_i \) is called the set of schedulability test points of \( \tau_i \).

Lemma 1 and Theorem 2 are related proofs under the mathematic induction procedure. We first prove that if any process set with an offspring set replaced with the corresponding RSR is schedulable, then the original process set is schedulable (i.e., Lemma 1 as an induction base). The proof is done by locating a feasible scheduling time point. We then show that the same feasible scheduling time point would exist for a process set with and without any number of offspring sets being replaced with their corresponding RSR’s (i.e., Theorem 2 as an induction hypothesis and an induction step).

For the readability, the proofs of most theorems, lemmas, and corollaries in this paper are included in the appendix.

**Lemma 1** Suppose that \( T_{i-1} \) is schedulable. Let \( \tau_j \) be any process in \( T_i, TO_j \) a subset of an offspring set of \( \tau_j \) in \( T_i \) which includes \( \tau_j \), and \( \tau \) the RSR of \( TO_j \). Let process \( \tau' \) be the process with the largest period in \( (T_i - TO_j \cup \{\tau\}) \), where the period and computation requirement of \( \tau' \) are \( p_i \) and \( c \), respectively. If there exists a pair \((k, m) \in R' \) such that

\[
\sum_{\tau_x \in (T_i - TO_j \cup \{\tau\} - \{\tau'\})} (c_x \left\lceil \frac{mp_x}{p_x} \right\rceil) + c \leq mp_k,
\]

where \( R' = \{(k, m) | \tau_k \in (T_i - TO_j \cup \{\tau\}), m = 1, 2, \cdots, \left\lfloor \frac{p_i}{p_k} \right\rfloor \} \), then \( T_i \) (including \( \tau_j \)) is schedulable.

**Proof.** Please see the appendix. □

We shall use the following example to illustrate the usage of the above lemma: Given a process set \( \{\tau_1, \tau_2, \tau_3\} \), let their computation requirements and the periods be \( \{(1,3),(1,5),(2,15)\} \), respectively. Suppose that \( T_2 = \{\tau_1, \tau_2\} \) is schedulable. Let \( TO_3 = \{\tau_2, \tau_3\} \) be an offspring set of \( \tau_3 \), and the RSR \( \tau = (5,15) \). If \( \{\tau_1, \tau\} \) is schedulable, then \( T_3 = \{\tau_1, \tau_2, \tau_3\} \) is schedulable.
**Theorem 2** Suppose that $T_{i-1}$ is schedulable, and $ST_i$ is a non-empty subset of $T_i$. For each process $\tau_j \in ST_i$, let $TO_j$ be a subset of an offspring set of $\tau_j$ in $T_i$ such that $TO_j \cap ST_i = \{\tau_j\}$, and $TO_j \cap TO_k = \emptyset$ for any two distinct processes $\tau_j$ and $\tau_k$ in $ST_i$. For each process $\tau_j \in ST_i$, $\tau'_j$ is the RSR of $TO_j$. Let process $\tau'$ be the process with the largest period in $T'_i = (T_i - \bigcup_{j \in ST_i} TO_j \cup \{\tau'_j|\tau'_j$ is the RSR of $TO_j, for every $\tau_j \in ST_i\})$, where the period and computation requirement of $\tau'$ are $p_i$ and $c$, respectively. If there exists a pair $(k, m) \in R'$ such that

$$\sum_{\tau_x \in (T'_i - \{\tau'\})} \left(c_x \left\lceil \frac{mp_k}{p_x} \right\rceil \right) + c \leq mp_k,$$

where $R' = \{(k, m)|\tau_k \in T'_i, m = 1, 2, \ldots, \left\lfloor \frac{p_i}{p_k} \right\rfloor\}$, then $T_i$ (including $\tau_i$) is schedulable.

**Proof.** Please see the appendix. $\square$

We shall use the following example to illustrate the usage of the above theorem: Given a process set $\{\tau_1, \tau_2, \tau_3\}$, let their computation requirements and the periods be $\{(1,3),(1,5),(2,15)\}$, respectively. Consider a newly arrived task $\tau_4 = (1,24)$. We could first partition $T_4$ into two disjoint offspring sets: $TO_3 = \{(1,5),(2,15)\}$ and $TO_4 = \{(1,3),(1,24)\}$, and let the RSR's of $TO_3$ and $TO_4$ be $(5,15)$ and $(9,24)$, respectively. As a result, if $\{(5,15),(9,24)\}$ is schedulable, then $\{(1,3),(1,5),(2,15),(1,24)\}$ is schedulable.

**Theorem 3** [22] A set of $n$ periodic processes is schedulable if the total utilization factor of the process set is no larger than $n(2^{1/n} - 1)$.

**Theorem 4** Suppose that $T_{i-1}$ is schedulable. Let $K$ be the number of roots in $T_i$. If the total utilization factor of $T_i$ is no larger than $K(2^{1/K} - 1)$, then $T_i$ is schedulable.

**Proof.** Please see the appendix. $\square$

Since the number of roots in $T_i$ is much less than the number of processes (and the number of fundamental frequencies) in $T_i$, we surmise that the utilization bound for a process set can be much improved. We can show as follows that Theorem 4 can justify the schedulability of a process set more precisely than Theorem 5 [13] and Theorem 3 [22] do.

**Theorem 5** [13] A set of periodic processes with $K$ fundamental frequencies is schedulable if the total utilization factor of the process set is no larger than $K(2^{1/K} - 1)$.

**Corollary 1** Let $T$ be a set of periodic processes. If $T$ is guaranteed to be schedulable according to Theorem 3, then $T$ is guaranteed to be schedulable according to Theorem 4.

**Proof.** Please see the appendix. $\square$
Corollary 2 Let $T$ be a set of periodic processes. If $T$ is guaranteed to be schedulable according to Theorem 5, then $T$ is guaranteed to be schedulable according to Theorem 4.

Proof. Please see the appendix. □

Corollary 3 There exists a process set which is guaranteed to be schedulable according to Theorem 4 but not according to Theorems 3 and 5.

Proof. Please see Example 2. □

3.1.3 Schedulability Test Algorithm

Theorem 4 suggests an iterative procedure to show the schedulability of a set $T$ of $n$ periodic processes: Let the total utilization factor of $T_i$ be $U_i$, where $T_i = \{\tau_1, \cdots, \tau_i\}$ is a subset of $T$ with processes whose periods are less than or equal to $p_i$.

Step 1: $i = 1$;
Step 2: If there are $K$ roots in $T_i$, and $U_i \leq K(2^{1/K} - 1)$,
Then $\tau_i$ is schedulable;
Else the schedulability of $\tau_i$ cannot be guaranteed; Exit;
Step 3: $i = i + 1$;
Step 4: Goto Step 2 unless $i > n$.

If the test fails when $i = k$, then only processes in $T_{k-1}$ are guaranteed to be schedulable! If the test completes successfully, all processes are schedulable. The complexity of the algorithm is $O(n^2)$. A comparison among the schedulability-tests based on process number, fundamental frequency number, and (division-graph) root number is given in the next example.

Example 2 Schedulability Tests:

Let $T$ be a set of five processes with timing constraints shown in Table 3, and $U_i = \sum_{j=1}^{i} \frac{c_j}{p_j}$ the utilization factor of process set $T_i = \{\tau_1, \cdots, \tau_i\}$. $T_1$, $T_2$, and $T_3$ are schedulable according to the utilization bound $U(n) = n(2^{1/n} - 1)$ proposed by Liu and Layland [22], where $n$ is the number of processes in $T_i$. However, the schedulability of $T_4$ and $T_5$ cannot be guaranteed by the Liu&Layland’s utilization bound because $U_4 = 0.8166 > U(4) = 0.7568$ and $U_5 = 0.95 > U(5) = 0.9114$.

<table>
<thead>
<tr>
<th>$\tau_i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_i$</td>
<td>3</td>
<td>5</td>
<td>15</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>$c_i$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>$U_i$</td>
<td>0.3333</td>
<td>0.5333</td>
<td>0.6666</td>
<td>0.8166</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 3: The period $p_i$ and computation requirement $c_i$ of each process
0.7435 (Please see Theorem 3). However, since there are only two fundamental frequencies, i.e., \( \{3, 15\} \) and \( \{5, 20\} \), the utilization bound of \( T_4 \) should be \( U(2) = 0.8284 \) according to Theorem 5, and \( T_4 \) (with \( U_4 = 0.8166 < U(2) \)) is schedulable. Unfortunately, the schedulability of \( T_5 \) still cannot be guaranteed by the Kuo&Mok’s utilization bound, i.e., the bound in Theorem 5, since there are two fundamental frequencies in \( T_5 \), i.e., \( \{3, 15\} \) and \( \{5, 20, 60\} \), and \( U_5 = 0.95 > U(2) = 0.83 \). However, since there is only one root in \( T_5 \), \( T_5 \) is guaranteed to be schedulable by Theorem 4 since \( U_5 = 0.95 \leq U(1) = 1 \). □

3.2 On-Line Incremental Test

We now present an incremental version of the schedulability test described in the previous section for on-line admission control.

Suppose that a system consists of a collection of \( N \) existing processes \( ES = \{ \tau_1 = (c_1, p_1), \cdots, \tau_n = (c_N, p_N) \} \) sorted in a non-decreasing order of their periods. Suppose these \( N \) processes have been admitted to the system and a new process \( \tau = (c, p) \) has just arrived. Note that the relative deadline of each process is its period. W.l.o.g., we assume all process periods are distinct.

Let \( ET(\tau_i) \) denote the division graph of those processes in \( ES \) with a period no larger than the period of \( \tau_i \). \( Root(\tau_i) \) and \( U(\tau_i) \) denote the set of roots in \( ET(\tau_i) \) and the sum of utilization factors \( c_i/p_i \) of processes in \( ET(\tau_i) \), respectively. Let \( j \) be the maximum index for \( \tau_j \) in \( ES \) whose period is less than the period \( p \) of Process \( \tau \). The following algorithm decides if the new process can be admitted to the system:

**Step 1:** \( Root(\tau) = Root(\tau_j) + \{p\} - \{p_k|p_k \in Root(\tau_j) \& p_k|p\} \);
**Step 2:** \( U(\tau) = U(\tau_j) + \frac{c}{p} \);
**Step 3:** If \( U(\tau) > K(2^{1/K} - 1) \), where \( K \) is the size of \( Root(\tau) \)

Then Return(REJECT);
**Step 4:** \( i = j + 1; \)
**Step 5:** ComputeRootFlag = True;
**Step 6:** While \( (i \leq N) \) Do
**Step 7:** If ComputeRootFlag

Then If \( (p \text{ does not divide } p_i) \)

Then \( Root(\tau_i) = Root(\tau_i) + \{p\} - \{p_k|p_k \in Root(\tau_i) \& p_k|p\} \);
Else ComputeRootFlag = False; /* \( Root(\tau_i) \) remains for the rest \( i/\)

**Step 8:** \( U(\tau_i) = U(\tau_i) + \frac{c_i}{p_i} \);
**Step 9:** If \( U(\tau_i) > K(2^{1/K} - 1) \), where \( K \) is the size of \( Root(\tau_i) \)

Return(REJECT);
**Step 10:** \( i = i + 1; \)
**Step 11:** EndWhile;
**Step 12:** Return(ACCEPT);
The schedulability of process $\tau$ and all existing processes only need to consider processes whose periods are no more than $p$. Steps 1 and 2 derive the number of roots of the corresponding division subgraph and the accumulated CPU utilization for processes with periods no more than $p$ (including Process $\tau$). The time complexity of the computation is proportional to the number of roots in $\text{Root}(\tau_j)$. The loop between Step 5 and Step 11 derives the number of roots of the corresponding division subgraph and the accumulated CPU utilization for processes with periods no more than $p_i$. The time complexity of each iteration is proportional to the number of roots in $\text{Root}(\tau_i)$. Variable $\text{ComputeRootFlag}$ is set to optimize the algorithm performance when the period of $p$ will no longer affect the set of roots in the while loop. The period of $p$ will no longer affect the set of roots in the while loop when $p$ can divide the period of some process.

The time complexity of the incremental algorithm is $\sum_{i=j}^{N} |\text{Root}(\tau_i)|$. Note that the number, i.e., $|\text{Root}(\tau_i)|$, of roots in each division subgraph is usually much less than the number of processes in the division subgraph (e.g., multimedia processes often have harmonic service rates). Although $|\text{Root}(\tau_i)|$ may approach $N$ in the worst case when no processes’ rate can divide each other, we surmise that the incremental algorithm will behave like a linear-time algorithm in general because process service rates in most real-time systems are often chosen from a small collection of values to simplify the system design and to improve the system performance.

For comparison, many other schedulability tests, such as those proposed in [9, 13], can not be used in an incremental way. In particular, the schedulability test based on the number of fundamental frequencies must be re-run, whenever a new process arrives, to derive the number of fundamental frequencies in a system [13]. The schedulability test based on the Sr and DCT period transformation methods [9] must also recalculate the special base of the entire process set, whenever a new process arrives, to minimize the increase of the system utilization factor during period transformation. The tests [9, 13] are also much more complicated than the proposed schedulability test in this section. Later in this paper, we will show that the schedulability test proposed in this paper outperforms any existing polynomial-time schedulability test [9, 13, 22].

### Example 3 Incremental Admission control:

We illustrate the incremental algorithm by an example. Consider a set of five existing processes.
as shown in Table 4. Their division graph is shown in Figure 1. Suppose a new process \( \tau = (c = 3, p = 30) \) arrives at the system. Processes \( \tau_1, \tau_2, \tau_3, \) and \( \tau_4 \) remain schedulable since their periods are shorter and thus the priorities are higher. The incremental algorithm first derives \( \text{Root}(\tau) = \text{Root}(\tau_4) + \{30\} - \{15\} = \{20, 30\} \) and \( \text{U}(\tau) = \text{U}(\tau_4) + \frac{3}{30} = 0.8 \). Since there are only two roots in \( \text{Root}(\tau) \), and \( \text{U}(\tau) = 0.8 \leq 2(2^{1/2} - 1) \), the algorithm enters the loop between Step 5 and Step 11 with \( i = 5 \). Since \( p, i.e., 30, \) divides \( p_5, i.e., 60, \) \( \text{Root}(\tau_5) \) remains the same. The algorithm then derives \( \text{U}(\tau_5) = \text{U}(\tau_5) + \frac{3}{60} = 0.9 \). Since there is only one root in \( \text{Root}(\tau_5) = \{60\} \), \( \text{U}(\tau_5) = 0.9 \leq 1(2^{1/1} - 1) = 1.0 \), the system increments \( i \) and returns “ACCEPT”. The new process is accepted, and all processes can be successfully guaranteed. □

4 Schedulability Test for the Multi-Frame Model

The purpose of this section is to extend our results to the multiframe process model for more precise efficient schedulability tests. A basic schedulability test and its incremental implementation are also presented. The proposed schedulability test was shown to outperform any other existing schedulability tests [10, 26, 27].

4.1 Motivation

We now extend our result in the last section to processes with varying timing requirements in different frames (or periods), such as multimedia sessions. One example is VCD movies that are often MPEG-1 encoded. MPEG-1 encoded video streams usually consist of a sequence of groups of pictures, and each group of pictures consists of 15 frames with the following pattern: I, B, B, P, B, B, P, B, B, P, B, B, I [20].

The multiframe model proposed by Mok and Chen [26, 27] provides a better analysis model to exploit the schedulability of processes with variations in their timing constraints. They showed that the proposed schedulability test is much more precise than the schedulability test proposed in [22], when the execution time of a process varies sharply from frame to frame [26, 27]. Unfortunately, the schedulability test proposed in [26, 27] does not consider the harmonic relationship of process periods. As a result, schedulable but heavily loaded process sets are difficult to be identified by the schedulability test proposed in [26, 27], where many computer systems, such as multimedia servers, are often heavily loaded. Note that any multiframe-based schedulability test based on period transformation methods, such as \( DCT \) and \( Sr \) in [9, 10], will also have difficulty in indentifying schedulable but heavily loaded process sets because of the same reason in the schedulability analysis of the Liu&Layland-based process set. The transformation methods
usually result in the sharp increase of processor utilization and seriously damage the preciseness of the proposed schedulability test at heavy or even moderate system load [18]¹.

In this section we will show our schedulability test that improves the test proposed by Mok and Chen [26, 27] by exploring the harmonic relationship of process periods. Our work can also be compared favorably to [10], as we will see from experiments later in this paper. In the following sections, we will present a much more precise schedulability test based on the multiframe process model, which will be shown in Section 5 to greatly outperform existing work [10, 26, 27], especially when the system is heavily loaded. We will then provide an incremental algorithm for on-line implementation.

4.2 Efficient Schedulability Test Based on the Multi-Frame Model

4.2.1 Definitions and Claim

In this section, we will first summarize important definitions from [26, 27] and define the reduced set for multiframe processes.

Definition 5 [26, 27] Multiframe Process:

A multiframe real-time process \( \tau_i \) is a tuple \((\Gamma_i, p_i)\), where \( \Gamma_i \) is an array of \( N_i \) execution times \((C^0_i, C^1_i, \cdots, C^{N_i-1}_i)\) for some \( N_i \geq 1 \), and \( p_i \) is the period of \( \tau_i \). Let \( C^0_i \) be the maximum in an array of execution times \((C^0_i, C^1_i, \cdots, C^{N_i-1}_i)\). \( C^0_i \) is called the peak execution time of \( \tau_i \).

Definition 6 [26, 27] AM Multiframe Process:

An array \((C^0_i, C^1_i, \cdots, C^{N_i-1}_i)\) is said to be AM (Accumulative Monotonic) if \( \sum_{k=0}^{j} C^k_i \mod N_i \geq \sum_{k=x}^{x+j} C^k_i \mod N_i, \ 1 \leq x \leq (N_i-1), \ 1 \leq j < (N_i-1) \). A multiframe process \( \tau_i = \{(C^0_i, C^1_i, \cdots, C^{N_i-1}_i), p_i\} \) is said to be AM if its array of execution times is AM.

Note that AM is restrictive for process models. However, it is critical for the worst-case schedulability analysis (e.g., critical-instant-based tests). In order to adopt the worst-case schedulability

¹The multimedia response-time analysis proposed by Sjodin and Hansson [38] is based on a more general multiframe model and rate-based scheduling disciplines. Authors adopt Weighted Fair Queuing (WFQ) and Calculus for Network Delays (CND) to provide liner equations (with several parameters) to bound the transmission of multimedia streams over ATM. The analysis also considers practical issues, such as jitters. The methodology proposed in our paper is for priority-based schedulability tests. Our objective is to provide a basic and theoretic foundation for more efficient (priority-based) schedulability tests. We must emphasize that our work must follow the AM restriction because of the considerations of critical instants.
analysis, engineers have to refine/modify the array of execution times for non-AM processes such that the revised execution time is no smaller than the corresponding execution time in the original array.

Lemma 2 [26, 27] The Critical Instance of AM Multiframe Processes:

The critical instance of an AM multiframe process is the beginning of the period when its peak execution time is requested simultaneously with the peak execution times of all higher priority processes.

Definition 7 Reduced Set and RSR of AM Multiframe Processes:

Suppose that \( \tau = \{ \Gamma = \{(C^0, C^1, \cdots, C^{N-1}), p\} \) is a multiframe periodic process with period \( p \) and an array \( \Gamma \) of computation requirements. A set of \( n \) AM multiframe periodic processes \( \{\tau_1, \cdots, \tau_i, \cdots, \tau_n\} \), where process \( \tau_i \) has period \( p_i \) and an array \( \Gamma_i = (C^0_i, C^1_i, \cdots, C^{N_i-1}_i) \) of \( N_i \) computation requirements, is a reduced set of \( \tau \) if \( p_i | p \), \( p = \max_{i=1}^n p_i \), \( N \) is the least common multiple of \( N_i \) for \( 1 \leq i \leq n \), and for all \( 0 \leq j \leq (N-1) \), \( C^j = \sum_{i=1}^n (\sum_{k=0}^{(p/p_i)-1} C^j_i (j/p_i + k) \mod N_i) \). \( \tau \) is called the RSR of the reduced set.

Note that the definitions of roots and offspring sets remain the same for multiframe process sets. For the readability of the paper, the proofs of most theorems, lemmas, and corollaries in this paper are included in the appendix.

Theorem 6 The RSR \( \tau = \{ \Gamma = (C^0, C^1, \cdots, C^{N-1}), p\} \) of a set of \( n \) AM multiframe periodic processes \( \{\tau_1, \cdots, \tau_i, \cdots, \tau_n\} \) is an AM multiframe process.

Proof. Please see the appendix. \( \Box \)

Example 4 The RSR of an AM Multiframe Process Set:

![Diagram](image-url)  

Figure 2: The RSR of a multiframe process set.
Let an AM multiframe process set consist of two processes \( \tau_1 = \{(2, 1, 1), 3\}, \tau_2 = \{(3), 6\} \).
The peak utilization factor of the process set is \( \frac{7}{6} \). \( \tau = \{(6, 6, 5), 6\} \) is the RSR of the process set and is an AM multiframe process, as shown in Figure 2. The peak utilization factor \( \tau \) is \( \frac{6}{6} = 1 \). Note that the peak utilization factor decreases. \( \square \)

For the purpose of this paper, we only consider AM multiframe process sets in this section. The results of this section can be used to derive the utilization bound for the general process model proposed in [27]. Note that the traditional process model proposed by Liu and Layland [22] is a special case of the (AM) multiframe model.

### 4.2.2 Root-Based Schedulability Test

Since Mok and Chen [26, 27] showed that the rate monotonic scheduling (RMS) algorithm [22] is optimal for multiframe process scheduling, the RMS algorithm is adopted in the following discussions. In this section, we are interested in preemptive fixed-priority scheduling. The deadline of a process request is the next request time.

**Theorem 7** [26, 27] *For the preemptive fixed priority scheduling policy, an AM multiframe process is schedulable if it is schedulable at its critical instance.*

Suppose that \( T = \{\tau_1, \cdots, \tau_n\} \) is a set of \( n \) AM multiframe periodic processes, where \( \tau_i = \{\Gamma_i = (C_i^0, C_i^1, \cdots, C_i^{N_i-1}), p_i\} \). Let \( T_i = \{\tau_1, \cdots, \tau_i\} \) be a subset of \( T \) with processes whose periods are less than or equal to \( p_i \). When we say that a process set \( TS \) is schedulable, it means that every process in \( TS \) is schedulable at its critical instant (i.e., within its critical interval). The schedulability test proposed in Theorem 1 is revised for AM multiframe processes as follows:

**Theorem 8** Process \( \tau_i \) in a set of AM multiframe periodic processes scheduled by a fixed priority-driven preemptive scheduling algorithm will always meet its deadline for all process phases iff there exists a pair \( (k, m) \in R_i \) such that

\[
\sum_{j<i} \left( \sum_{x=0}^{\left\lceil \frac{mp_k}{p_j} \right\rceil - 1} C_j^x \text{ mod } N_j \right) + C_i^0 \leq mp_k,
\]

where \( R_i = \{(k, m)|1 \leq k \leq i, m = 1, 2, \cdots, \lfloor \frac{p_k}{p_i} \rfloor \} \).

**Proof.** Please see the appendix. \( \square \)

Lemma 3 and Theorem 9 are related proofs under the mathematic induction procedure. We first prove that if any process set with an offspring set replaced with the corresponding RSR is
satisfactory, then the original process set is schedulable (i.e., Lemma 3 as an induction base). The proof is done by locating a feasible scheduling time point. We then show that the same feasible scheduling time point would exist for a process set with and without any number of offspring sets being replaced with their corresponding RSR’s (i.e., Theorem 9 as an induction hypothesis and an induction step).

**Lemma 3** Suppose that the AM multiframe process set \( T_{i-1} \) is schedulable. Let \( \tau_j \) be any multiframe process in \( T_i \), \( TO_j \) a subset of an offspring set of \( \tau_j \) in \( T_i \) which includes \( \tau_j \), and \( \tau = \{ (C^0, C^1, \ldots, C^{N-1}), p \} \) the RSR of \( TO_j \). Let process \( \tau' \) be the process with the largest period in \( (T_i - TO_j \cup \{ \tau \}) \), where the period and the array of computation requirements of \( \tau' \) are \( p_i \) and \( \Gamma' = (C^0, C^1, \ldots, C'^{N'-1}) \), respectively. If there exists a pair \( (k, m) \in R' \) such that

\[
\sum_{\tau_x \in (T_i - TO_j \cup \{ \tau \})} \left( \sum_{y=0}^{\left\lfloor \frac{mp_k}{px} \right\rfloor - 1} C^y_x \mod N_x \right) + C^0 \leq mp_k,
\]

where \( R' = \{(k, m)|\tau_k \in (T_i - TO_j \cup \{ \tau \}), m = 1, 2, \ldots, \left\lfloor \frac{px}{mp_k} \right\rfloor \} \), then the AM multiframe process set \( T_i \) (including \( \tau_i \)) is schedulable.

**Proof.** Please see the appendix. □

**Theorem 9** Suppose that the AM multiframe process set \( T_{i-1} \) is schedulable, and \( ST_i \) is a non-empty subset of the AM multiframe process set \( T_i \). For each process \( \tau_j \in ST_i \), let \( TO_j \) be a subset of an offspring set of \( \tau_j \) in \( T_i \) such that \( TO_j \cap ST_i = \{ \tau_j \} \), and \( TO_j \cap TO_k = \phi \) for any two distinct processes \( \tau_j \) and \( \tau_k \) in \( ST_i \). For each process \( \tau_j \in ST_i \), \( \tau'_j \) is the RSR of \( TO_j \). Let process \( \tau' \) be the process with the largest period in \( T'_i = (T_i - \bigcup_{j \in ST_i} TO_j \cup \{ \tau'_j | \tau'_j \text{ is the RSR of } TO_j \}, \) for every \( \tau_j \in ST_i \} \), where the period and the array of computation requirements of \( \tau' \) are \( p_i \) and \( \Gamma' = (C'^0, C'^1, \ldots, C'^{N'-1}) \), respectively. If there exists a pair \( (k, m) \in R' \) such that

\[
\sum_{\tau_x \in (T'_i - \{ \tau' \})} \left( \sum_{y=0}^{\left\lfloor \frac{mp_k}{px} \right\rfloor - 1} C^y_x \mod N_x \right) + C^0 \leq mp_k,
\]

where \( R' = \{(k, m)|\tau_k \in T'_i, m = 1, 2, \ldots, \left\lfloor \frac{px}{mp_k} \right\rfloor \} \), then the AM multiframe process set \( T_i \) (including \( \tau_i \)) is schedulable.

**Proof.** Please see the appendix. □

**Definition 8** Peak Utilization Factor:

The peak utilization factor \( U_m \) of an AM multiframe process set \( T = \{ \tau_1, \ldots, \tau_n \} \) is equal to \( \sum_{i=1}^{n} C^0_i / p_i \), where \( C^0_i \) and \( p_i \) are the peak execution time and period of \( \tau_i \), respectively.
Theorem 10 \cite{26, 27} Let \( r = \min_{i=1}^{n}(C_0^i/C_1^i) \). For process sets of size \( n \), the bound on peak utilization factor is given by \( r \cdot n \cdot (\frac{r+1}{r}^{1/n} - 1) \).

Theorem 11 Suppose that \( T_{i-1} \) is schedulable. Let \( K \) be the number of roots in \( T_i \). If the total peak utilization factor of \( T_i \) is no larger than \( r \cdot K \cdot (\frac{r+1}{r}^{1/K} - 1) \), then \( T_i \) is schedulable, where \( r \) is calculated based on the RSR set of \( T_i \).

Proof. Please see the appendix. \( \square \)

Note that there is no way to compare the bounds on peak utilization factor derived by Theorems 10 and 11 because \( r \) might be different in the original and transformed process sets. However, Theorem 9 does show the capability of the root-based approach. We surmise that the bound on peak utilization factor derived by Theorem 11 should be much better than that derived by Theorem 10 on average because the number of roots is much less than the number of processes. In Section 5, a series of experiments was conducted to demonstrate the capability of the root-based approach, for which we have very encouraging experimental results.

Note that the iterative algorithm for schedulability test proposed in Section 3 can be applied here with Theorem 11. However, the complexity of the algorithm becomes \( O(K \cdot N^2) \), where \( K \) is the maximum size of the array of execution times for multiframe processes during the creation of RSRs.

4.3 On-Line Implementation

We now present an incremental version of the schedulability test described in the previous section for on-line admission control. The incremental algorithm for the Liu&Layland model proposed in Section 3 can be revised as follows:

Suppose that a system consists of a collection of \( N \) existing multiframe processes \( ES = \{\tau_1 = (\Gamma_1 = (C_0^1, C_1^1, \ldots, C_{N-1}^1), p_1), \ldots, \tau_n = (\Gamma_N = (C_0^N, C_1^N, \ldots, C_{N-1}^N), p_N)\} \) sorted in a non-decreasing order of their periods. Let the \( N \) existing processes be schedulable and operating. The question is whether to admit a new process \( \tau = (\Gamma = (C_0^\tau, C_1^\tau, \ldots, C_{M-1}^\tau), p) \). For the simplicity of algorithm presentation, we assume that process periods are distinct (the proposed algorithm can be extended in a straightforward way when processes may have the same period.)

Let \( ET(\tau_i) \) denote the subgraph of the division graph of processes in \( ES \) which have a period no larger than the period of \( \tau_i \), and \( Root(\tau_i), U(\tau_i), \) and \( RSRep(\tau_i) \) denote the set of roots in \( ET(\tau_i) \), the sum of utilization factors \( c_i/p_i \) of processes in \( ET(\tau_i) \), and a set of the RSRs in \( ET(\tau_i) \), respectively. Note that the set of the periods of the RSRs in \( ET(\tau_i) \) is equal to \( Root(\tau_i) \).
The schedulability of all existing processes and the new process is guaranteed (with the rate-
monotonic scheduling [22]) if the following algorithm returns “ACCEPT”: Let \( j \) be the maximum
index for \( \tau_j \) in \( ES \) whose period is less than the period \( p \) of Process \( \tau \). Suppose the array of the
computation requirement of \( \tau \) is \( \Gamma = (C_0, C_1, \ldots, C^{M-1}) \).

**Step 1:** If there exists some \( p_k \in \text{Root}(\tau_j) \) which can divide \( p \)
Then \( \text{RSRep}(\tau) \) is created by merging the corresponding RSRs in \( \text{RSRep}(\tau_j) \) with \( \tau \);
Else \( \text{RSRep}(\tau) = \text{RSRep}(\tau_j) + \{\tau\} \);

**Step 2:** \( \text{Root}(\tau) = \) the set of periods of RSRs in \( \text{RSRep}(\tau) \);

**Step 3:** If \( \text{RSRep}(\tau) \) is not schedulable according to Theorem 11
Then Return(REJECT);

**Step 4:** \( i = j+1 \);

**Step 5:** While \( (i \leq N) \) Do

**Step 6:** \( \text{RSRep}(\tau_i) \) is created by merging the corresponding RSRs in \( \text{RSRep}(\tau_j) \) with \( \tau \);

**Step 7:** \( \text{Root}(\tau_i) = \) the set of periods of RSRs in \( \text{RSRep}(\tau_i) \);

**Step 8:** If \( \text{RSRep}(\tau_i) \) is not schedulable according to Theorem 11
Then Return(REJECT);

**Step 9:** \( i = i + 1 \);

**Step 10:** EndWhile;

**Step 11:** Return(ACCEPT);

The multiframe-based incremental algorithm is the same as the Liu&Layland-based incremen-
tal algorithm proposed in Section 3, except that the set of RSRs must be maintained. As a
result, the time complexity of the incremental algorithm becomes \( \sum_{i=j}^N K[\text{Root}(\tau_i)] \), where \( K \) is
the maximum size of the array of execution times for multiframe processes dring the creation
of RSRs. Note that the number, i.e., \( \text{Root}(\tau_i) \), of roots in each division subgraph is much less
than the number of processes in the division subgraph, and many processes in a system, such as
multimedia servers or control systems, are often harmonically related. For example, \( K \) is usually
restrictively bounded in many multimedia systems where video streams are often encoded in a
similar way, e.g., VCD (or MPEG-1) encoded. We surmise that the incremental algorithm will
be extremely efficient and behave like a linear-time algorithm in general.

Note that the multiframe-based schedulability test based on the Sr and DCT period transfor-
mation methods [10] has a time complexity \( O(N^2(N + \max N_i)) \), where \( N_i \) is the size of the
array of execution times of process \( \tau_i \). The schedulability test must re-run by recalculating the
special base of the entire process set, whenever a new process arrives, to minimize the increase
of the system utilization factor during period transformation [10]. The tests [10] are also much
more complicated than the proposed schedulability test in this section. In Section 5, we will
show that the proposed schedulability test significantly outperforms existing polynomial-time
schedulability tests [10, 26, 27], especially when the system is heavily loaded.
Table 5: The Root(), Peak U(), and RSRep() of existing processes

Example 5 Incremental Admission control:

We illustrate the incremental algorithm by an example. Consider a set of five existing processes, as shown in Table 5. Their division graph is shown in Figure 1. Let a new process $\tau = (\Gamma = (3, 2), p = 30)$ arrive. Processes $\tau_1, \tau_2, \tau_3$, and $\tau_4$ remain schedulable. The incremental algorithm first derives $RSRep(\tau) = \{(2, 1), 20\}, ((21, 20), 30\}$. Since $r = \min_{i=1}^{n} (C^0_i/C^1_i) = 1.05$, and the peak $U(\tau) = \text{peak} U(\tau_4) + \frac{3}{30} = 0.8$ is no more than the bound, i.e., 0.834, required by Theorem 11, the algorithm enters the loop between Step 5 and Step 11 with $i = 5$. The algorithm then derives $RSRep(\tau_5) = \{(52, 48), 60\}$. Since $r = \min_{i=1}^{n} (C^0_i/C^1_i) = 1.083$, and the peak $U(\tau_5) = \text{peak} U(\tau_5) + \frac{3}{30} = 0.9$ is no more than the bound, i.e., 1.0, required by Theorem 11, the system increments $i$ and returns “ACCEPT”. The new process is accepted, and all processes can be successfully guaranteed. □

5 Performance Evaluation

To assess the capability of the different schedulability tests in predicting the schedulability of a process set, we have conducted some experiments. There are two parts in the experiments. The first part compares the capability of different schedulability tests based on the Liu and Layland process model [22], i.e., the schedulability tests based on the number of processes [22] (i.e., Theorems 3 and 10), the number of fundamental frequencies [13] (i.e., Theorem 5), and the number of roots (i.e., Theorems 4 and 11), and the polynomial-time schedulability tests $DCT + Sr$ proposed by Han and Tyan [9]. The second part compares the capability of the proposed multiframe schedulability test with the tests proposed by Mok and Chen [26, 27] and Han [10].

5.1 Data Set and Measurement

The primary performance metric used is the ratio of process sets guaranteed to be schedulable by each schedulability test, referred to as Guarantee Ratio. Let $n$ be the number of process sets randomly generated in the simulation, and $m$ the number of process sets guaranteed to be schedulable by a schedulability test. The Guarantee Ratio of the schedulability test is $\frac{m}{n}$.
The process sets for performance evaluation were based on benchmarks and systems reported in [41, 39, 40, 42]. In particular, the Hartstone [41] benchmark aimed at the measuring of the effectiveness and the efficiency of real-time schedulers and their systems. Response time was a major performance metric. The benchmark considered two kinds of workloads: harmonic or non-harmonic workloads. Under harmonic workloads, process periods were 2, 4, or 8 times of a specified base period. Under non-harmonic workloads, process periods were 3, 5, or 7 times of a specified base period. In other words, the maximum number of fundamental frequencies (/roots) was 3. The systems reported in [39, 40, 42] were based on realistic workloads. In those studies, the ratio of the number of fundamental frequencies and the number of processes was between $\frac{1}{4}$ and $\frac{1}{6}$ of the number of processes in the process set.

We adopted a random number generator to generate process sets: For the experiments of the Liu&Layland process model, the number of processes per process set was randomly chosen between 10 and 30. The number of fundamental frequencies were basically within two ranges: They were $[\frac{1}{10}, \frac{1}{3}]$ and $[\frac{1}{4}, \frac{1}{16}]$ of the number of processes in the process set. The experiments started with process sets of a utilization factor equal to 70% and increased by 5% at a time until the utilization factor was equal to 95%. 400 process sets per utilization factor were tested. The utilization factor of each process was no more than 20% of the total utilization factor of the process set. Each process was assigned fundamental frequencies randomly, and the possibility of assigning $i$ fundamental frequencies to a process was $\frac{1}{2^i-1}$.

For the experiments of the multiframe process model, the number of processes per process set was randomly chosen between 10 and 15, and the number of fundamental frequencies were within a range: That was $[\frac{1}{1}, \frac{1}{16}]$ of the number of processes in the process set. The experiments started with process sets of a peak utilization factor equal to 70% and increased by 5% at a time until the peak utilization factor was equal to 110%. 400 process sets per peak utilization factor were tested. The utilization factor of each process was no more than 20% of the total utilization factor of the process set. Each process $\tau_i$ was assigned a period $p_i$ and an array of $N_i$ execution times $(C_i^0, C_i^1, \cdots, C_i^{N_i-1})$ as follows: Let $\tau_i$ and $\tau_j$ be two distinct processes sharing a fundamental frequency, and $i < j$. $\tau_i$ was assigned a period which divided the period of $\tau_j$. The periods of processes which did not share any fundamental frequency with other processes or only share some fundamental frequencies with a larger-index process were randomly chosen in the range $(7, 29)$ (they are prime numbers). The size $N_i$ of the execution time array $(C_i^0, C_i^1, \cdots, C_i^{N_i-1})$ of each multiframe process was randomly chosen in the range $(1, 20)$. $C_i^0$ was derived based on the period and peak utilization factor of $\tau_i$. $C_i^j = C_i^0 \frac{1}{r_i}$ for $0 \leq j \leq (N_i - 1)$, and $r_i$ was randomly chosen in the range $(2, 5)$. 
5.2 Simulation Results for the Liu&Layland Process Model

Figure ?? and Figure ?? show the guarantee ratios of different schedulability test algorithms when the number of fundamental frequencies were within two ranges: They were $\left[ \frac{1}{4}, \frac{1}{10} \right]$ and $\left[ \frac{1}{4}, \frac{1}{10} \right]$ of the number of processes in a process set, respectively. In Figure ?? and Figure ??, it was observed that the guarantee ratio of the root-based schedulability test proposed in this paper greatly outperformed the schedulability tests based on the number of processes [22] and the number of fundamental frequencies [13] and the $DCT + Sr$-based schedulability tests [9], especially when the utilization factor of a process set is beyond 75%. (The guarantee ratio of the schedulability test based on the number of processes is zero when the utilization factor of a process set is beyond 70%).

Note that the average number of roots and the average number of fundamental frequencies in Figure ?? were 2.13 and 4.80, respectively. The average number of roots and the average number of fundamental frequencies in Figure ?? were 1.41 and 2.85, respectively. That explained why the root-based schedulability test proposed in this paper greatly outperformed the schedulability tests based on the number of processes [22] and the number of fundamental frequencies [13]. As astute readers might notice, the schedulability tests based on the number of roots and the number of fundamental frequencies [13] had better performance when the ratio of the number of fundamental frequencies and the number of processes was lower (i.e., Figure ??). We should also point out that the guarantee ratio of the root-based tests decreased significantly when the utilization factor increased from 80% to 85%. It was because many process sets which had two or more roots would fail the schedulability test since $2 \cdot (2^{\frac{1}{2}} - 1) = 0.8284$.

The difference between the guarantee ratios of the root-based and $DCT + Sr$-based schedulability test becomes very significant when the utilization factor of a process set is over 75%, which is not even considered as a heavy load. It is because the increasing amount of the utilization factor in the process-set transformation for the $DCT + Sr$ schedulability test becomes very significant for process sets with a large utilization factor [18]. The experimental results reveal the weakness of period-transform methods such as $DCT$ and $Sr$ [9, 18]. The performance of the $DCT + Sr$-based schedulability test quickly deteriorates when the system becomes slightly heavily loaded. On the other hand, the root-based schedulability test performs consistently well and greatly outperforms the $DCT + Sr$-based schedulability test [9] and any other tests when the utilization factor of a process set is above 75%.

When the root-based schedulability test and $DCT + Sr$ schedulability test [9] are used jointly, the guarantee ratio improves only slightly. When the utilization factor of a process set is over 85%, the $DCT + Sr$-based schedulability test [9] fails in identifying any schedulable process sets.
Figure 3: The guarantee ratio of Liu&Layland, Kuo&Mok, $DCT + Sr$-based, and root-based schedulability tests. (The ratio of the number of fundamental frequencies and the number of processes was between $\frac{1}{1}$ and $\frac{1}{4}$.)

It shows that although the root-based schedulability test and $DCT + Sr$ schedulability test [9] complement each other (when the system is lightly loaded), the $DCT + Sr$ schedulability test does not help much in identifying any schedulable process sets which could not be identified by the root-based schedulability test.

The reason why the root-based schedulability tests only had a small improvement over other approaches, when the utilization factor was not high (i.e., under 80%), was because the utilization factors of experimented process sets were often no larger than the values of the corresponding values of $k \cdot (2^\frac{k}{k} - 1)$, regardless of whether $k$ was the number of roots, the number of fundamental frequencies, or the number of processes. For example, the values of $k \cdot (2^\frac{k}{k} - 1)$ were 83%, 78%, 75.7%, 74.3%, and 73.4%, when $k = 2, 3, 4, 5,$ and 6, respectively. When the utilization factor was over 80%, the root-based schedulability tests, that set $k$ as the number of roots in $k \cdot (2^\frac{k}{k} - 1)$, greatly outperformed other approaches because other approaches that set $k$ as the number of fundamental frequencies or the number of processes in $k \cdot (2^\frac{k}{k} - 1)$ would usually had $k \cdot (2^\frac{k}{k} - 1)$ being less than 80%. Note that the average number of roots in the experiments was only between 1.41 and 2.13, where the average number of fundamental frequencies was between 2.85 and 4.8.

The efficiency of the root-based algorithm was done by measuring the execution time of different schedulability tests proposed in the literature. The experiments were done on a single-board-computer (SBC) which equips with an Intel 80486 DX2-66 microprocessor to reflect the specs of many embedded systems currently in the market. We run the schedulability tests over
Table 6: The execution time of the proposed efficient schedulability tests on 486-DX2 66MHz, where the CPU utilization was 80%. Note that the two schedulability tests proposed by other researchers did not provide on-line incremental algorithms.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Han &amp; Tyan</td>
<td>12.5ms</td>
</tr>
<tr>
<td>Kou &amp; Mok</td>
<td>0.775ms</td>
</tr>
<tr>
<td>Root-based</td>
<td>0.65ms</td>
</tr>
</tbody>
</table>

400 process sets and averaged their execution times. The summary is shown in Table 6. The values in the first column were to measure the execution times of the schedulability tests over an entire process set, where the second column measured those for on-line incremental admission control, as shown in Section 3.2. To compare the execution times of efficient schedulability tests with those of precise schedulability tests, e.g., the Rate Monotonic Analysis (RMA) [19, 33], we run the RMA schedulability tests over many process sets. It was observed that the required execution time of exact schedulability tests on Pentium III-800MHz would be many seconds and sometime went up to thousands of seconds.

5.3 Simulation Results for the Multiframe Process Model

Figure ?? shows that the guarantee ratio of the root-based schedulability test greatly outperforms the schedulability tests proposed by Mok and Chen [26, 27] (i.e., results labeled as “Mok & Chen”) and Han [10] (i.e., results labeled as “Mok & Chen +DCT + Sr”), when $r_i$ (which was the ratio of $C_i^0$ and $C_i^1$) is randomly chosen in the range (2, 5). Many more process sets are successfully identified to be schedulable by the RMS scheduling algorithm. It is because the number of roots is much smaller than the number of processes. In the simulation experiments, we also observe that the average $r = \min_{i=1}^{n}(C_i^0/C_i^1)$ even increases in the set of the RSRs of offspring sets of roots in each process set. It also made contribution to the performance of the root-based schedulability test. The average $r$ in the root-based schedulability test is 2.441, while the average $r$ in the original process sets is about 2.015. On the other hand, the $DCT + Sr$-based schedulability test [10] fails in identifying any schedulable process sets, when the peak utilization factor of a process set is close to 90%. It is because the increasing amount of the utilization factor in the process-set transformation for the $DCT + Sr$ schedulability test becomes very significant for process sets with a large utilization factor [18].

The peak utilization factors of many multiframe process sets also decrease in the root-based schedulability test because the variation of per-period execution time of a RSR often decrease.
in the root-based schedulability test (due to the merging of processes). That explains why the root-based schedulability test can guarantee the schedulability of many process sets with a peak utilization factor over 100%. Example 4 presented earlier in the paper may help in explaining this phenomenon.

When the upper bound of $r = \min_{i=1}^{n} (C_i^0/C_i^1)$ in data generation was increased from 5 to 10, Figure ?? shows that the guarantee ratio of the root-based schedulability test also greatly outperforms the schedulability tests proposed by Mok and Chen [26, 27] and Han [10], when $r_i$ was randomly chosen in the range $(2, 10)$. The average $r$ in the root-based schedulability test is 3.329, while the average $r$ in the original process sets is about 2.135.

## 6 Conclusions

This paper proposes efficient on-line schedulability tests which consider not only the harmonic relationship of process periods but also the variance of computation times in different periods. The proposed schedulability tests are shown to be much more precise and efficient than existing polynomial-time schedulability tests [2, 9, 10, 13, 22, 26, 27]. Unlike many other polynomial-time schedulability tests, the proposed schedulability tests perform consistently well, regardless of whether the system is lightly or heavily loaded! We also provide an incremental implementation of the schedulability tests for efficient on-line admission control. The experimental results clearly show the merits of this methodology. A comparison of different schedulability tests is summarized in Table 7.
Table 7: The effectiveness and efficiency of schedulability test algorithms. Note that the schedulability tests proposed by other researchers did not provide on-line incremental algorithms.

With the advance of software and hardware technologies, there is an increasing demand to study the real-time resource allocation problems for different architectural assumptions and computation models. For future research, we will extend the reduced-set methodology to analyze the schedulability of “mixed” real-time systems consisting of hard, soft, and firm real-time processes and explore different guarantee levels for different kinds of processes. We will also further explore the application semantics of multimedia systems to improve the capacity of the systems in resource allocation.

Acknowledgements

We would like to thank Ching-Huei Li for his help in our simulation experiments.

References


Lemma 1 Suppose that $T_{i-1}$ is schedulable. Let $\tau_j$ be any process in $T_i$, $TO_j$ a subset of an offspring set of $\tau_j$ in $T_i$ which includes $\tau_j$, and $\tau$ the RSR of $TO_j$. Let process $\tau'$ be the process with the largest period in $(T_i - TO_j \cup \{\tau\})$, where the period and computation requirement of $\tau'$ are $p_i$ and $c$, respectively. If there exists a pair $(k, m) \in R'$ such that

$$\sum_{\tau_x \in (T_i - TO_j \cup \{\tau\})} (c_x \left\lceil \frac{m p_k}{p_x} \right\rceil) + c \leq m p_k,$$

where $R' = \{(k, m) | \tau_k \in (T_i - TO_j \cup \{\tau\}), m = 1, 2, \cdots, \left\lceil \frac{p_x}{p_k} \right\rceil\}$, then $T_i$ (including $\tau_j$) is schedulable.

Appendix: Proofs of Theorems, Lemmas, and Corollaries
Proof. Suppose that there exists a pair \((k, m) \in R'\) such that
\[
\sum_{\tau_x \in (T_i - TO_j) \cup \{\tau'\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + c \leq mp_k,
\]
where \(R' = \{(k, m) | \tau_k \in (T_i - TO_j) \cup \{\tau\}, m = 1, 2, \ldots, \left\lfloor \frac{p_x}{p_k} \right\rfloor\}\), we will show as follows that \(mp_k\) is a scheduling time point at which \(\tau_i\) is schedulable. Note that \(\tau'\) is \(\tau\) if \(\tau_j\) is \(\tau_i\). If \(\tau_j\) is not \(\tau_i\), then \(\tau'\) is \(\tau_i\).

If \(\tau_j\) is \(\tau_i\), then \(\tau'\) is \(\tau_i\) and \(TO_j\) includes \(\tau_i\). By the definition of RSR, the utilization factor \(\frac{c}{p_j}\) of \(\tau\) is equal to \(\sum_{\tau_y \in TO_j} \frac{c_y}{p_y}\). The above formula for the scheduling time point \(mp_k\) can be re-written as follows:
\[
mp_k \geq \sum_{\tau_x \in (T_i - TO_j) \cup \{\tau\} \cup \{\tau'\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + c
\]
\[
= \sum_{\tau_x \in (T_i - TO_j) \cup \{\tau\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + \sum_{\tau_y \in TO_j} (c_y \frac{p_x}{p_y}) + c_i / \text{since } \tau_i \in TO_j
\]
\[
\geq \sum_{\tau_x \in (T_i - TO_j) \cup \{\tau\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + \sum_{\tau_y \in (T_i - \{\tau\})} (c_y \left\lceil \frac{mp_k}{p_y} \right\rceil) + c_i / \text{since } p_i \geq mp_k \text{ and } p_y | p_i
\]
\[
= \sum_{\tau_x \in T_{i-1}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + c_i
\]
If \(\tau_j\) is not \(\tau_i\), then \(\tau'\) is \(\tau_i\). The period of \(\tau\) is \(p_j\). By the definition of RSR, the utilization factor \(\frac{c}{p_j}\) of \(\tau\) is equal to \(\sum_{\tau_y \in TO_j} \frac{c_y}{p_y}\). The above formula for the scheduling time point \(mp_k\) can be written as follows:
\[
mp_k \geq \sum_{\tau_x \in (T_i - TO_j) \cup \{\tau\} \cup \{\tau'\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + c_i / \text{Note that } \tau' = \tau_i.
\]
\[
= \sum_{\tau_x \in (T_i - TO_j) \cup \{\tau\} \cup \{\tau'\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + \frac{mp_k}{p_x} c_i + c_i
\]
\[
= \sum_{\tau_x \in (T_i - TO_j) \cup \{\tau\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + \frac{mp_k}{p_x} (p_j \sum_{\tau_y \in TO_j} \frac{c_y}{p_y}) + c_i
\]
\[
= \sum_{\tau_x \in (T_i - TO_j) \cup \{\tau\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + \sum_{\tau_y \in TO_j} (c_y \frac{p_x}{p_y} \left\lceil \frac{mp_k}{p_y} \right\rceil) + c_i
\]
\[
= \sum_{\tau_x \in (T_i - TO_j) \cup \{\tau\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + \sum_{\tau_y \in TO_j} (c_y \left\lceil \frac{mp_k}{p_y} \right\rceil) + c_i / \text{since } p_y | p_j
\]
\[
\geq \sum_{\tau_x \in (T_i - TO_j) \cup \{\tau\}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + \sum_{\tau_y \in (T_i - \{\tau\})} (c_y \left\lceil \frac{mp_k}{p_y} \right\rceil) + c_i / \text{since } |\{\tau\}| \geq |\{\tau\}|
\]
\[
= \sum_{\tau_x \in T_{i-1}} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + c_i
\]
Regardless of whether \(\tau_j\) is \(\tau_i\), \(mp_k\) is proven as a scheduling time point at which \(\tau_i\) is schedulable (Please see Theorem 1). Since \(T_{i-1}\) is schedulable, and \(\tau_i\) is schedulable, \(T_i\) is schedulable. □

**Theorem 2** Suppose that \(T_{i-1}\) is schedulable, and \(ST_i\) is a non-empty subset of \(T_i\). For each process \(\tau_j \in ST_i\), let \(TO_j\) be a subset of an offspring set of \(\tau_j\) in \(T_i\) such that \(TO_j \cap ST_i = \{\tau_j\}\), and \(TO_j \cap TO_k = \phi\) for any two distinct processes \(\tau_j\) and \(\tau_k\) in \(ST_i\). For each process \(\tau_j \in ST_i\), \(\tau'_j\) is the RSR of \(TO_j\). Let process \(\tau'\) be the process with the largest period in \(T'_i = (T_i - \bigcup_{j \in ST_i} TO_j \cup \{\tau'_j | \tau'_j \text{ is the RSR of } TO_j, \text{for every } \tau_j \in ST_i\})\), where the period and computation requirement of \(\tau'\) are \(p_i\) and \(c\), respectively. If there exists a pair \((k, m) \in R'\) such that
\[
\sum_{\tau_x \in (T'_i - \{\tau'\})} (c_x \left\lceil \frac{mp_k}{p_x} \right\rceil) + c \leq mp_k,
\]
where \( R' = \{(k, m)|\tau_k \in T'_i, m = 1, 2, \ldots, \lfloor \frac{p_i}{p_k} \rfloor \} \), then \( T_i \) (including \( \tau_i \)) is schedulable.

**Proof.** The proof can be done by applying the same argument adopted in Lemma 1 to every RSR \( \tau'_j \) in \( ST_i \).

**Theorem 4** Suppose that \( T_{i-1} \) is schedulable. Let \( K \) be the number of roots in \( T_i \). If the total utilization factor of \( T_i \) is no larger than \( K(2^{1/K} - 1) \), then \( T_i \) is schedulable.

**Proof.** Since \( T_i \) can be entirely and exclusively partitioned into subsets \( TO_j \) of the offspring sets of the \( K \) roots \( Root_j \), \( T_i \) can be transformed into a process set \( T'_i \) which include the \( K \) RSRs of the \( TO_j \)'s. Because the utilization factor of \( T'_i \) is no larger than \( K(2^{1/K} - 1) \), the process \( \tau' \) with the largest period in \( T'_i \) is schedulable according to Theorem 3. Note that the utilization factors of \( T_i \) and \( T'_i \) are the same according to the definition of RSR. Based on Theorem 1, there must exist a pair \((k, m) \in R'\) such that

\[
\sum_{\tau_x \in (T'_i-(\tau'))} (c_x \left\lfloor \frac{mp_k}{p_x} \right\rfloor) + c \leq mp_k,
\]

where \( R' = \{(k, m)|\tau_k \in T'_i, m = 1, 2, \ldots, \lfloor \frac{p_i}{p_k} \rfloor \} \), where the period and computation requirement of \( \tau' \) are \( p_i \) and \( c \), respectively. The correctness of this proof then follows directly from Theorem 2.

**Corollary 1** Let \( T \) be a set of periodic processes. If \( T \) is guaranteed to be schedulable according to Theorem 3, then \( T \) is guaranteed to be schedulable according to Theorem 4.

**Proof.** The proof is trivial because the number of roots is no larger than the number of processes.

**Corollary 2** Let \( T \) be a set of periodic processes. If \( T \) is guaranteed to be schedulable according to Theorem 5, then \( T \) is guaranteed to be schedulable according to Theorem 4.

**Proof.** The proof is trivial because the number of roots is no larger than the number of fundamental frequencies, and the set of processes in a fundamental frequency is a special case of an offspring set of some process in the fundamental frequency.

**Theorem 6** The RSR \( \tau = \{\Gamma = (C^0, C^1, \ldots, C^{N-1}), p\} \) of a set of \( n \) AM multiframe periodic processes \( \{\tau_1, \ldots, \tau_i, \ldots, \tau_n\} \) is an AM multiframe process.

**Proof.** The AM property of \( \tau \) can be proven by showing \( \sum_{k=0}^{j} C^k \mod N \geq \sum_{k=x}^{x+j} C^k \mod N \), \( 1 \leq x \leq (N-1), 1 \leq j < (N-1) \).
The formula in the first line 1 could be re-written as that in the second line because \( j < N - 1 \).

It could be further re-written as that in the third line because we have \( C^j = \sum_{i=1}^{n} (\sum_{k=0}^{(p/p_i)-1} C_i^{(k(p/p_i)+y)} \mod N_i) \) according to Definition 7. The fourth line could be re-written as that in the fifth line when the rightmost two sigma equations were expanded. The \( \geq \) relationship holds because \( \tau_1, \tau_2, \ldots, \tau_n \) are AM. The derivation of the formulae of seventh, eighth, and ninth lines could be done in similar ways as those for third, fourth, and fifth lines \( \square \)

**Theorem 8** Process \( \tau_i \) in a set of AM multiframe periodic processes scheduled by a fixed priority-driven preemptive scheduling algorithm will always meet its deadline for all process phases iff there exists a pair \((k,m) \in R_i\) such that

\[
\sum_{j<i} (\sum_{x=0}^{\lfloor \frac{m p_k}{p_i} \rfloor - 1} C_j^x \mod N_j) + C_i^0 \leq m p_k,
\]

where \( R_i = \{(k,m)|1 \leq k \leq i, m = 1, 2, \ldots, \lfloor \frac{p_i}{p_k} \rfloor\} \).

**Proof.** The correctness of this theorem directly follows from Theorems 1 and 7. Note that, at every schedulability test point of \( R_i \), new computation requests from higher-priority processes may arrive. \( \square \)

**Lemma 3** Suppose that the AM multiframe process set \( T_{i-1} \) is schedulable. Let \( \tau_j \) be any multiframe process in \( T_i \), \( TO_j \) a subset of an offspring set of \( \tau_j \) in \( T_i \) which includes \( \tau_j \), and \( \tau = \{ (C^0, C^1, \ldots, C^{N-1}), p \} \) the RSR of \( TO_j \). Let process \( \tau' \) be the process with the largest period in \( (T_i - TO_i \cup \{ \tau \}) \), where the period and the array of computation requirements of \( \tau' \) are \( p_i \) and \( \Gamma' = (C^0, C^1, \ldots, C^{iN'-1}) \), respectively. If there exists a pair \((k,m) \in R'\) such that

\[
\sum_{\tau_x \in (T_i-TO_i \cup \{ \tau \})-\{ \tau' \}} (\sum_{y=0}^{\lfloor \frac{m p_k}{p_x} \rfloor - 1} C_x^y \mod N_x) + C^0 \leq m p_k,
\]
where \( R' = \{(k, m) | \tau_k \in (T_i - TO_j \cup \{ \tau \}) \}, m = 1, 2, \ldots, \left[ \frac{p_i}{p_j} \right] \}, \) then the AM multiframe process set \( T_i \) (including \( \tau_i \)) is schedulable.

**Proof.** Suppose that there exists a pair \( (k, m) \in R' \) such that

\[
\sum_{\tau \in (T_i - TO_j \cup \{ \tau \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x + C^0 \right) \leq mp_k,
\]

where \( R' = \{(k, m) | \tau_k \in (T_i - TO_j \cup \{ \tau \}) \}, m = 1, 2, \ldots, \left[ \frac{p_i}{p_j} \right] \}, \) we will show as follows that \( mp_k \) is a scheduling time point at which \( \tau_i \) is schedulable. Note that \( \tau' \) is \( \tau \) if \( \tau_j \) is \( \tau_i \). If \( \tau_j \) is not \( \tau_i \), then \( \tau' \) is \( \tau_i \).

If \( \tau_j \) is \( \tau_i \), then \( \tau' \) is \( \tau_i \), and \( TO_j \) includes \( \tau_i \). \( C^k = C^{ik} \) for \( 0 \leq k \leq N \). By the definition of RSR, \( C^j = \sum_{\tau \in TO_j} \sum_{k=0}^{(p_i/p_j)-1} C^y_z (j(p_i/p_j)+k \mod N_z) \), for \( 0 \leq j \leq (N-1) \). The above formula for the scheduling time point \( mp_k \) can be written as follows:

\[
mp_k \geq \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x + C^0 \right)
\]

\[
= \sum_{\tau \in (T_i \cup \{ \tau \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{\tau \in TO_j} \left( \sum_{k=0}^{(p_i/p_j)-1} C^y_z (k \mod N_z) \right)
\]

\[
= \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{\tau \in TO_j - \{ \tau_i \}} \left( \sum_{k=0}^{(p_i/p_j)-1} C^y_z (k \mod N_z) \right) + C^0 \]

\[
/ \star \text{since } \tau_i \in TO_j \}/ \]

\[
\geq \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{\tau \in TO_j - \{ \tau_i \}} \left( \sum_{k=0}^{[\frac{mpk}{p_j}]-1} C^y_z (k \mod N_z) \right) + C^0 \]

\[
/ \star \text{since } p = p_i, p_i \geq mp_k, \text{ and } p_z | p_j \}/ \]

\[
= \sum_{\tau \in T_{i-1}} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + C^0 \]

If \( \tau_j \) is not \( \tau_i \), then \( \tau' \) is \( \tau_i \). The period \( p \) of \( \tau \) is \( p_j \). By the definition of RS-representative, \( C^m = \sum_{\tau \in TO_j} \sum_{k=0}^{(p_i/p_j)-1} C^y_z (m(p_i/p_j)+k \mod N_z) \), for \( 0 \leq m \leq (N-1) \). The above formula for the scheduling time point \( mp_k \) can be written as follows:

\[
mp_k \geq \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + C^0 \]

\[
= \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N + C^0 \]

\[
= \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N + C^0 \]

\[
= \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N + C^0 \]

\[
= \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N + C^0 \]

\[
= \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N + C^0 \]

\[
= \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N + C^0 \]

\[
\geq \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N + C^0 \]

\[
\geq \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N + C^0 \]

\[
\geq \sum_{\tau \in (T_i - TO_j \cup \{ \tau \} - \{ \tau' \})} \left( \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N_x \right) + \sum_{y=0}^{[\frac{mpk}{p_j}]-1} C^y_x \mod N + C^0 \]
Regardless of whether \( \tau_j \) is \( \tau_i \), \( mp_k \) is proven as a scheduling time point at which \( \tau_i \) is schedulable (Please see Theorem 8). Since \( T_{i-1} \) is schedulable, and \( \tau_i \) is schedulable, \( T_i \) is schedulable. \( \Box \)

**Theorem 9** Suppose that the AM multiframe process set \( T_{i-1} \) is schedulable, and \( ST_i \) is a non-empty subset of the AM multiframe process set \( T_i \). For each process \( \tau_j \in ST_i \), let \( TO_j \) be a subset of an offspring set of \( \tau_j \) in \( T_i \) such that \( TO_j \cap ST_i = \{ \tau_j \} \), and \( TO_j \cap TO_k = \phi \) for any two distinct processes \( \tau_j \) and \( \tau_k \) in \( ST_i \). For each process \( \tau_j \in ST_i \), \( \tau'_j \) is the RSR of \( TO_j \). Let process \( \tau' \) be the process with the largest period in \( T'_i = (T_i - \bigcup_{j \in ST_i} TO_j) \cup \{ \tau'_j | \tau'_j \text{ is the RSR of } TO_j \text{, for every } \tau_j \in ST_i \} \), where the period and the array of computation requirements of \( \tau' \) are \( p_i \) and \( \Gamma' = (C^{o_0}, C^{o_1}, \cdots, C^{o_{N'-1}}) \), respectively. If there exists a pair \((k, m) \in R'\) such that

\[
\sum_{\tau_k \in T'_i} (\sum_{y=0}^{\lfloor \frac{mp_k}{px} \rfloor -1} C^y_x \mod N_z) + C^0_i \leq mp_k,
\]

where \( R' = \{(k, m)|\tau_k \in T'_i, m = 1, 2, \cdots, \lfloor \frac{ps}{pk} \rfloor \} \), then the AM multiframe process set \( T_i \) (including \( \tau_i \)) is schedulable.

**Proof.** The proof can be done by applying the same argument adopted in Lemma 3 to every RSR \( \tau'_j \) in \( ST_j \). \( \Box \)

**Theorem 11** Suppose that \( T_{i-1} \) is schedulable. Let \( K \) be the number of roots in \( T_i \). If the total peak utilization factor of \( T_i \) is no larger than \( r \cdot K \cdot (r+1)^{1/K} - 1 \), then \( T_i \) is schedulable, where \( r \) is calculated based on the RSR set of \( T_i \).

**Proof.** Since \( T_i \) can be entirely and exclusively partitioned into subsets \( TO_j \) of the offspring sets of the \( K \) roots \( \text{Root}_{j} \), \( T_i \) can be transformed into a process set \( T'_i \) which include the \( K \) RSRs of the \( TO_j \)’s. Because the utilization factor of \( T'_i \) is no larger than \( r \cdot K \cdot (r+1)^{1/K} - 1 \), the process \( \tau' \) with the largest period in \( T'_i \) is schedulable according to Theorem 10. Based on Theorem 8, there must exist a pair \((k, m) \in R'\) such that

\[
\sum_{\tau_x \in T'_i} (c_x \lfloor \frac{mp_k}{px} \rfloor) + c \leq mp_k,
\]
$R' = \{(k, m)|\tau_k \in T'_i, m = 1, 2, \ldots, \lfloor \frac{p_k}{p_i} \rfloor\}$, where the period and computation requirements of $\tau'$ are $p_i$ and $c$, respectively. The correctness of this proof then follows directly from Theorem 9. □