A Precise Schedulability Test Algorithm for Scheduling Periodic Tasks in Real-Time Systems

Wan-Chen Lu  
National Tsing Hua University  
Hsinchu 300, Taiwan, R.O.C.  
wlu@cs.nthu.edu.tw

Jen-Wei Hsieh  
National Taiwan University  
Taipei 106, Taiwan, R.O.C.  
d90002@csie.ntu.edu.tw

Wei-Kuan Shih  
National Tsing Hua University  
Hsinchu 300, Taiwan, R.O.C.  
wshih@cs.nthu.edu.tw

ABSTRACT
Rate monotonic analysis (RMA) has been shown to be effective in the schedulability analysis of various types of system. This paper focuses on reducing the run time of each RMA-tested system. Based on a new concept of tasks, denoted by the lift-utilization tasks, we propose a novel method to reduce the number of iterative calculations in the derivation of the worst-case response time of each task in its RMA test. The capability of the proposed method was evaluated and compared to related work, which revealed that our method produced savings of 26–33% in the number of RMA iterations.

Keywords
Real-Time Systems, Schedulability Test, Rate Monotonic Analysis, Periodic Tasks, Fixed Priority Preemptive Scheduling

1. INTRODUCTION
Feasibility and schedulability problems have received considerable attention in the real-time systems research community in recent decades. Various sufficient conditions for testing the schedulability of priority-based task sets have been proposed (e.g., [6, 8]), and excellent work on precise tests has also been presented (e.g., [2, 9]). The sufficient conditions provide closed-form (but imprecise) tests with polynomial-time complexities. However, they also give a more intuitive insight into the effect of changing certain timing parameters on the system performance, especially when real-time tasks are admitted dynamically.

Whilst precise tests may provide better schedulability guarantees for priority-driven systems, many of them require the task response time to be calculated iteratively in the presence of higher priority tasks. Lehoczky et al. [9] were among the first to propose the precise-test concept for rate monotonic analysis (RMA), in which a set of time points is verified. In the verification process, we need to check if the total computation for a given task set before a time point (which is in the set) can be completed before the time point. This RMA method is referred to as the time-demand analysis method. Audsley et al. [2] proposed another original idea for the RMA, in which the worst-case response time $W_R_i$ of each task $\tau_i$ is derived by iteratively calculating the formula $W_{R_i}^{(l+1)} = c_i + \sum_{k=1}^{l} W_{R_k}^{(l)} \times c_i$ until $W_{R_i}^{(l+1)}$ either converges to a real number (i.e., $W_R_i$) or continues beyond the deadline of task $\tau_i$, where tasks are assumed to be sorted in increasing order of priority, and $W_{R(0)} = \sum_{i=1}^{n} c_i$. In the remainder of this paper, this RMA method is referred to as the worst-case simulation method (WCSM). In recent years, researchers have proposed several excellent methods to improve the run time of RMA tests (e.g., [3, 5]). In particular, Bril et al. [5] proposed a better way to derive the initial value $W_{R(0)}$ for the WCSM for each task. Bini and Buttazzo [3] proposed a way to balance the required run time of the time-demand analysis method and the false-identification rate of schedulable tasks.

This paper studies the WCSM and provides an alternative version thereof. In each iteration of a WCSM test, a time instant $W_{R(i+1)}$ is obtained to derive the accumulated computation time up to the time instant $W_{R(i)}$. However, the time demand as calculated by the WCSM is too conservative, such that the performance of the WCSM is not satisfied. In this paper we present the concept of $\alpha$-WCSM and lift-utilization (LU) task set to solve this problem. The capability of the proposed method was evaluated and compared to related work (e.g., [2, 5]), which revealed that our method produced savings of 26–33% in the number of RMA iterations.

The remainder of this paper is organized as follows. In Section 2 we define some terminology and provide a motivation for the proposed method. Section 3 presents a fast algorithm for WCSM tests, and proves the correctness of the algorithm. Section 4 provides the simulation results from our evaluation of the capability of the proposed method, and Section 5 presents our conclusions and future work.

2. DEFINITIONS AND MOTIVATION
A periodic task comprises an infinite sequence of jobs with regular arrival times, where the task is a template of its corresponding jobs and one job of the task arrives at the beginning of each period. Let $T = \{\tau_1 = (c_1, p_1), \tau_2 = (c_2, p_2), \ldots, \tau_n = (c_n, p_n)\}$ be a set of periodic tasks, where $c_i$ and $p_i$ denote the maximum computation time and the period of a task $\tau_i$, respectively.
c_i and p_i are the maximum computation time and the period of task \( \tau_i \), respectively. Without loss of generality, we assume that \( p_1 \leq p_2 \leq \cdots \leq p_n \). Let \( T_n \) be scheduled by the rate monotonic scheduling algorithm (RM) [8], since RM is an optimal fixed-priority scheduling algorithm where the individual task priorities are inversely proportional to their respective periods. The priority of \( \tau_i \) is larger than the priority of \( \tau_{i+1} \) for \( 1 \leq i \leq (n - 1) \). We use \( T_{n-1} \) to denote the subset of \( T_n \); that is, \( T_{n-1} = \{ \tau_1, \tau_2, \ldots, \tau_{n-1} \} = T_n - \{ \tau_n \} \). To analyze the schedulability of a task set, a schedulability test is applied on \( T_i \), iteratively, for \( i = 1, \ldots, n \), to check whether the lowest-priority task in \( T_i \) (i.e., \( \tau_n \) under RM) is schedulable. To facilitate the discussion in the remainder of this paper, we assume that tasks in \( T_{n-1} \) are all schedulable and confine our attention to verifying whether the lowest-priority task in \( T_n \) (i.e., \( \tau_n \) under RM) is schedulable. We first address some definitions.

**Definition 1. Utilization Factor [8]:**

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

where \( c_i \) and \( p_i \) are the maximum computation time and the period of task \( \tau_i \), respectively; \( \frac{c_i}{p_i} (- = u_i) \) is called the utilization factor of \( \tau_i \).

**Definition 2. \( \alpha \)-WCSM:** For a periodic task set \( T_n \), let \( \alpha \) be the portion of processor time available for real-time tasks in each time unit. We define the \( \alpha \)-WCSM – which is a modification of the original WCSM – as follows:

\[
WR^{(i+1)} = \frac{U \alpha + \sum_{k=1}^{n-1} \frac{WR^{(i)}}{p_k} \times c_k}{\alpha}
\]

The utilization of the processor is partitioned into two parts, where the basic utilization is for executing the LU tasks and the WCSM utilization is for executing the remaining tasks. The LU task is formally defined as follows:

**Definition 3. LU tasks:** If a periodic task \( \tau_i \) is classified as an LU task in the \( l \)-th iteration of a WCSM test, then the total computation requirements of jobs for \( \tau_i \) before a time instant \( t \) is set to \( t \times u_i \), which is less than the actual total computation requirements of jobs for \( \tau_i \) before a time instant \( t \).

It should be noted that once a task is added to the set of LU tasks, the basic utilization is lifted by the utilization of this task and the processor needs to spend more time on the LU tasks. The behavior of the LU task is regular and predictable. Since we will not overestimate the time demand of LU tasks, the concept of the original WCSM still works well on the remaining tasks, but the utilization of the processor is reduced to \( \alpha \).

A WCSM test can be revised as follows based on the idea of the \( \alpha \)-WCSM and the LU tasks defined above:

1. Tasks are partitioned into two sets: \( L \) (including all LU tasks defined in Definition 3) and \( R = T_n - L \).
2. Let \( \alpha = 1 - \sum_{\tau_i \in L} u_i \); this means that the proportion of each time unit consumed by all LU tasks in \( L \) is \( 1 - \alpha \) (i.e., \( \sum_{\tau_i \in L} u_i \)). The utilization left to execute tasks in \( R \) is \( \alpha \).
3. Apply the \( \alpha \)-WCSM (defined in Definition 2) to \( R \) and find the next checking time instant

\[
WR'^{(l+1)} = \frac{\sum_{\tau_j \in R} \frac{WR^{(l-1)}}{p_j} \times c_j}{\alpha} = \frac{\sum_{\tau_j \in R} \frac{WR^{(l-1)}}{p_j} \times c_j}{1 - \sum_{\tau_j \in L} u_i}.
\]

The reason for naming those tasks as “lift-utilization” tasks is clear from the concept of \( \alpha \)-WCSM, since LU tasks increase the basic utilization consumed by all LU tasks. When we move one task from the WCSM task set to LU-task set, the utilization of WCSM tasks is decreased, but the number of tasks in the WCSM task set is also reduced by one. Therefore, how to partition tasks into LU and WCSM tasks represents a trade-off.

### 3. AN ALGORITHM FOR WCSM TESTS

The rationale behind the proposed algorithm, namely the adaptive-WCSM (A-WCSM) algorithm, is to attempt to have a larger jump in the derivation of each subsequent time instant \( WR^{(i)} \). The technical question is how to identify an appropriate LU task set in each iteration of a WCSM test so as to maximize the jump in each derivation of \( WR^{(i)} \). To be more specific, an LU task set, denoted as \( L(i) \), must be identified in each \( l \)-th iteration. \( R(i) \) (i.e., \( T_n - L(i) \)) denotes the set of the remaining tasks, where \( T_n \) is the set of tasks under a WCSM test. Given a task set \( T_n = \{ \tau_1, \tau_2, \ldots, \tau_n \} \) (sorted in decreasing order of priority) under a WCSM test, we assume that \( \tau_1, \tau_2, \ldots, \tau_{n-1} \) are all schedulable. We shall focus on the schedulability test for \( \tau_n \). A WCSM test should find the earliest time instant \( WR'_n \) in the scheduling of \( T_n \) at which the processor becomes idle for the first time. Note that we use \( WR'_n \) to denote the worst-case completion time of \( \tau_n \) under the A-WCSM algorithm. \( WR'_n \) denotes the time instant derived in the \( l \)-th iteration of the A-WCSM algorithm. When there is no ambiguity, \( WR'_n \) is denoted as \( WR^{(i)} \) since we are mainly interested in the schedulability test for \( \tau_n \). (\( WR^{(i)} \) denotes the time instant derived in the \( l \)-th iteration of the original WCSM test. Note that \( WR'_n \) is the same as the worst-case completion time of \( \tau_n \) derived in the original WCSM test.) With slight abuse of the notation, for simplicity we denote \( L(i) \) and \( R(i) \) in the \( l \)-th iteration by \( L \) and \( R \) in the statements of the A-WCSM algorithm.

#### 3.1 The A-WCMS Algorithm

The A-WCSM algorithm is shown in Algorithm 1. \( WR^{(i)} \) is set to \( \sum_{j=1}^{n} c_i \) [1], and each iteration derives a new value of \( WR^{(i)} \). When \( WR^{(l-1)} \) is found, we can calculate \( x_i = \frac{WR^{(l-1)}}{p_i} \times p_i \) for each task \( \tau_i \). A cut point, denoted by \( q \), is then selected to partition the tasks in \( T_n \) into \( L \) and \( R \). In our algorithm, the cut point selected is \( q = WR^{(l-1)} + Ratio \times diff \). Task \( \tau_i \) with \( x_i \) less than \( q \) is added to \( L \). Note that the variable \( Ratio \) is set to 0.5 for now, which is an intuitive positive real number for the checking (however, it can be replaced by any positive real number). In Section 4, we show the results for several values of \( Ratio \), from which we suggest a better value. \( WR^{(i)} \) is calculated based on the equation

\[
WR^{(l)} = \frac{\sum_{\tau_j \in R} \frac{WR^{(l-1)}}{p_j} \times c_j}{1 - \sum_{\tau_j \in L} u_i}
\]

where \( R = T - L \). If \( WR^{(l)} \) is less than or equal to \( WR^{(l-1)} \), \( WR^{(l)} \) is reset to \( WR^{(l-1)} \), and \( WR^{(l+1)} \) is
found by considering all the tasks that are in R. If \( WR' (l) \) has passed the deadline for \( r_n \), then \( r_n \) is unschedulable. If \( WR' (l) \) is equal to \( WR' (l-1) \), then \( WR' (l) \) converges to a real number, and \( r_n \) is schedulable; otherwise, the next iteration in the calculation of \( WR' (l+1) \) begins.

Algorithm 1 Adaptive-WCSM(\( \tau_n \))

Require: \( \tau_1, \tau_2, \ldots, \tau_{n-1} \) are all schedulable and \( \tau_n \) is the task for the checking

1: \( WR'(0) = \sum_{i=1}^{n} c_i \)
2: \( Ratio = 0.5 \) (this can be replaced by any positive real number)
3: \( diff = WR'(0) \)
4: \( l = 0 \)
5: repeat
6: \( l = l + 1 \)
7: \( LiftUtil = 0, TimeDemand = 0 \)
8: \( q = WR'(l-1) + Ratio \times diff \)
9: for \( i = 1 \) to \( q \) do
10: \( x_i = \left\lceil \frac{WR'(l-1)}{p_i} \right\rceil \times p_i \)
11: if \( x_i < q \) then
12: \( LiftUtil = LiftUtil + x_i \)
13: else
14: \( TimeDemand = TimeDemand + \left\lceil \frac{WR'(l-1)}{p_i} \right\rceil \times c_i \)
15: end if
16: end for
17: \( WR'(l) = \frac{TimeDemand}{1 - LiftUtil} \)
18: if \( WR'(l) \leq WR'(l-1) \) then
19: \( WR'(l) = WR'(l-1) \)
20: \( l = l + 1 \)
21: \( WR'(l) = \sum_{i=1}^{n} \left\lceil \frac{WR'(l-1)}{p_i} \right\rceil \times c_i \) (put all tasks in \( T_n \) into \( R \))
22: end if
23: \( diff = WR'(l-1) - WR'(l) \)
24: if \( WR'(l) > p_n \) then
25: report("unschedulable");
26: end if
27: if \( WR'(l) = WR'(l-1) \) then
28: report("schedulable");
29: end if
30: until \((WR'(l) = WR'(l-1)) \vee (WR'(l) > p_n))\)

We illustrate our algorithm with two examples:

Example 1. For a task set \( T = \{ \tau_1 = (2.4), \tau_2 = (1.5), \tau_3 = (3.3, 15) \} \), assume that all tasks are ready at time 0. The operation of our algorithm is as follows: \( WR'(0) \) is set to 6.3 (\( \sum_{i=1}^{3} c_i \)). By partitioning \( T_n \) into \( L(1) = \{ \tau_1 \} \) and \( R(1) = \{ \tau_2, \tau_3 \} \), we can obtain the next time instant \( WR'(1) = \frac{6.3}{1 - 0.5} = 10.6 \), as shown in Fig.1(a). The next time instant \( WR'(2) \) is derived by partitioning \( T_n \) into subsets \( L(2) = \{ \tau_1 \} \) and \( R(2) = \{ \tau_2, \tau_3 \} \), for which \( WR'(3) = 12.6 \), as shown in Fig.1(b). \( WR'(3) \), which is equal to 14.3, is found by partitioning \( T_n \) into \( L(3) = \emptyset \) and \( R(3) = \{ \tau_1, \tau_2, \tau_3 \} \), as shown in Fig.1(c). \( WR'(4) \) is made equal to 13.79 by setting \( L(4) = \{ \tau_2, \tau_3 \} \) and \( R(4) = \{ \tau_1 \} \). Because \( WR'(4) \) is less than \( WR'(3) \), \( WR'(4) \) is reset to \( WR'(3) \) (14.3). Then, \( WR'(5) \) is found by setting \( L(5) = \emptyset \) and \( R(5) = \{ \tau_1, \tau_2, \tau_3 \} \).

Since \( WR'(4) = WR'(5) = 14.3 \), the termination condition is satisfied and our algorithm reports that the task set is “schedulable”.

Example 1 illustrates the details of our algorithm, whereas Example 2 demonstrates the efficiency of our algorithm, and uses a task set for which our algorithm can achieve a large improvement in the number of iterations for the RMA schedulability test compared with the WCSM. It is very interesting that our approach finishes the test in near constant time in this example.

Example 2. For a task set \( T = \{ \tau_1 = (1.6, 2), \tau_2 = (0.76, 4), \tau_3 = (3, 301) \} \), assume that all tasks are ready at time 0. \( WR'(0) \) is set to 5.36 (\( \sum_{i=1}^{3} c_i \)). We can partition \( T_n \) into \( L(1) = \{ \tau_1, \tau_2 \} \) and \( R(1) = \{ \tau_3 \} \) and obtain the next time instant \( WR'(1) = \frac{5.36}{1 - 0.5} = 10.72 \). The next time instant \( WR'(2) \), which is equal to 0, is derived by partitioning \( T_n \) into \( L(2) = \{ \tau_1, \tau_2, \tau_3 \} \) and \( R(2) = \emptyset \). Because \( WR'(2) \) is less than \( WR'(1) \), \( WR'(2) \) is reset to \( WR'(1) \) (\( = 10.72 \)). Then, \( WR'(3) \) is also made equal to 300 by partitioning \( L(3) = \emptyset \) and \( R(3) = \{ \tau_1, \tau_2, \tau_3 \} \). Since \( WR'(2) = WR'(3) \), the termination condition is met and our algorithm reports that the task set is “schedulable”. Hence, the schedulability test is finished in four iterations. If the WCSM is applied to the same task set, the number of required iterations increases to 117 (at time instants 5.36, 9.32, \ldots, 2984.300, 300). This indicates that our algorithm significantly outperforms the WCSM for this task set.

3.2 Properties

The correctness of our algorithm is proved by the following lemmas and theorems.

Lemma 1. For every time instant derived by the A-WCSM...
algorithm, during each iteration (denoted by $WR^{(t)}$) there is no idle time in the time interval $(0, WR^{(t)})$.

Proof. Suppose the A-WCSM algorithm terminates at time instant $WR^{(t)}$. We will prove the lemma by induction on $WR^{(0)}, WR^{(1)}, \ldots, WR^{(t)}$.

Induction Basis. For $t = 0$, $WR^{(0)} = \sum_{j=1}^{n} c_j$. It is obvious that $(0, \sum_{j=1}^{n} c_j)$ does not have idle time.

Induction Hypothesis. We assume that our lemma is true for the time instant $WR^{(k)}$, that is, there is no idle time in the time interval $(0, WR^{(k)})$. We need to prove that this is also true for the time instant $WR^{(k+1)}$, that is, there is no idle time in the time interval $(0, WR^{(k+1)})$.

Induction Step. We prove this induction step by contradiction. In our algorithm, $WR^{(k)} \leq WR^{(k+1)}$. Assume that there is idle time in $(WR^{(k)}, WR^{(k+1)})$. Without loss of generality, let $t_{idle}$ be the first idle time in $(WR^{(k)}, WR^{(k+1)})$. Since $t_{idle}$ is the first idle time in the time interval $(WR^{(k)}, WR^{(k+1)})$, there is no idle time in the time interval $(0, WR^{(k+1)})$, we can conclude that there is no idle time in the time interval $(0, t_{idle})$, where $WR^{(k)} \leq t_{idle} < WR^{(k+1)}$. All jobs that are ready before time $t_{idle}$ should be finished before $t_{idle}$, since otherwise $t_{idle}$ will not be an idle time. Let $L(k+1)$ be the set of LU tasks in the $(k+1)$-th iteration when we compute $WR^{(k+1)}$ at time $WR(k)$. Since $t_{idle}$ is an idle time in the RM schedule, $\sum_{\tau_j \in L} \left( t_{idle} - p_j \right) \times c_j = t_{idle}$. So, $\sum_{\tau_j \in L} \left( t_{idle} - p_j \right) \times c_j + \sum_{\tau_j \in L} \left( t_{idle} \times c_j \right) = t_{idle}$. It is obvious that $\sum_{\tau_j \in L} \left( t_{idle} \times u_i \right) + \sum_{\tau_j \in L} \left( t_{idle} \times c_j \right) \leq t_{idle}$, that is, $WR^{(k+1)} \leq t_{idle}$. This is a contradiction.

According to the induction step, the induction goes through, and hence our lemma is correct.

Lemma 2. When the A-WCSM algorithm reports that the task $T_n$ is “schedulable”, $T_n$ is indeed schedulable.

Proof. We shall prove this lemma by contradiction. Let $WR^{(r)}$ be the derived time instant when the A-WCSM algorithm reports “schedulable” in the $r$-th iteration but where there is no idle time in the time interval $WR^{(r)}, WR^{(r+1)}$ in the RM schedule (here $WR^{(r)}$ means some time instant later than $WR^{(r)}$). According to the A-WCSM algorithm, the variable diff will approach 0. When the value of diff is sufficiently small, we will eventually put all tasks into $R(r)$ and leave $L(r) = \Phi$, which means that no tasks are LU tasks. Therefore, we have $WR^{(r+1)} = \sum_{\tau_j \in R(r)} \left( \frac{WR^{(r)} \times c_j}{p_j} \right) = \sum_{\tau_j \in \{r_1, \ldots, r_n\}} \left( \frac{WR^{(r)} \times c_j}{p_j} \right) > WR^{(r)}$. This means that at least one new job of some task is ready at time $WR^{(r)}$. Therefore, if we execute the repeat-until loop in the A-WCSM algorithm one more time, we will find a new time point that is larger than or equal to $WR^{(r+1)}$. This is a contradiction to the “schedulable” termination condition of the A-WCSM algorithm.

Theorem 1. Assume that $T_n-1$ is schedulable. $T_n$ is schedulable if and only if the A-WCSM algorithm reports that it is “schedulable”.

Proof. ($\Rightarrow$) If $T_n$ is schedulable, there would be an idle time in the time interval $(0, p_n)$. According to Lemma 2, our algorithm will terminate and report “schedulable”.

($\Leftarrow$) If $T_n$ is unschedulable, there would be no idle time in the time interval $(0, p_n)$. According to Lemma 2, our algorithm cannot find an idle time. That is, the time instant will keep on increasing until going past $p_n$, and our algorithm will report that it is “unschedulable”.

4. PERFORMANCE EVALUATION

4.1 Experimental Setup, Data Sets, and Metrics

In this section we assess the proposed RMA-iteration speedup by comparing the A-WCSM algorithm with other RMA-based algorithms (i.e., the WCSM [2] and the initial-value RMA schedulability test (IVRS) [5]) in terms of the number of RMA iterations needed for each algorithm, where each RMA iteration derives $WR^{(t)}$ based on the result of $WR^{(t-1)}$. The primary performance metric was the improvement ratio of the proposed A-WCSM schedulability test relative to these algorithms, referred to as the Improvement Ratio. To consider the WCSM (the IVRS) in the comparison, let $x$ and $y$ be the number of RMA iterations for the A-WCSM algorithm and the WCSM (the IVRS) in running a schedulability test for 10,000 task sets, respectively. The Improvement Ratio was defined as $(y - x)/y$.

The task sets for performance evaluation were generated based on benchmarks and systems reported previously [7, 10]. We adopted a random number generator to generate task sets: the number of tasks per task set was randomly chosen between 10 and 30. The number of fundamental frequencies was a real number within a range $[1, 1]$ of the number of tasks in the task set. Since a task set with a utilization factor of no more than 69% would be feasible according to the Liu and Layland bound [8], the data sets for the experiments were generated with a utilization factor of between 75% and 100%. The experiments started with task sets with a utilization factor equal to 75%, and were repeated for sets with the utilization factor increasing in increments of 5% until it reached 100%. The utilization factor of each task was no more than 20% of the total utilization of its task set. Each task was assigned fundamental frequencies randomly, where the possibility of assigning fundamental frequencies to a task was $(1/2) + 1$. The period of each task was derived by the multiplication of each of its assigned fundamental frequencies. A total of 10,000 task sets were tested for each utilization factor, and their results were averaged.

4.2 Experimental Results

Each RMA-based algorithm was evaluated experimentally in the same way. The schedulability of each task in a task set was checked in a nonincreasing order of task priority. The schedulability test $LL(i)$ proposed by Liu and Layland was first used to verify the schedulability of tasks until some task failed this test. If the $i$-th task failed the $LL(i)$ test, then the schedulability of all remaining tasks (with an index of no less than $i$) were verified by the RMA-based algorithms in the experiments. In each experiment, the initial value $WR_0^{(0)}$ of the WCSM was set to $\sum_{i=1}^{n} c_i$, and the initial value $WR_0^{(0)}$ of the IVRS was set to $\max(c_i/(1 - \sum_{i=1}^{n} u_j), WR_{i-1} + c_i)$. Note that a task set was considered schedulable if all tasks in the set were schedulable.

Figures 2 and 3 show the Improvement Ratios of the A-WCSM algorithm (for different settings of Ratio in the A-WCSM algorithm) relative to the WCSM and the IVRS,
respectively. As shown in Figure 2, a better Improvement Ratio was achieved when the utilization factor of a task set was large, which is attributable to the A-WCSM algorithm tending to jump over many more schedulability testing points in such cases, compared to the other algorithms. Figure 3 shows the RMA-iteration Improvement Ratios for the A-WCSM algorithm relative to the IVRS. The Improvement Ratios were significant for all task sets irrespective of the utilization. However, the Improvement Ratios decreased when the utilizations of the task sets were over 95%, which is due to the formula used to set the initial value \( WR_i \) of the IVRS. Figure 3 shows that the improvement Ratios were over 26% when the value of \( Ratio \) in the A-WCSM algorithm was set to 0.2. The Improvement Ratio even reached 33% when the utilization of task sets was 95%. The results from the above simulation indicate that our A-WCSM algorithm represents a significant improvement over the comparison algorithms.

5. CONCLUSIONS AND FUTURE WORK

This paper proposes the concept of LU tasks, and applies it to reducing the number of iterative calculations in the derivation of the worst-case response time of each task in its WCSM test. The proposed algorithm is denoted by the A-WCSM algorithm, in which we dynamically select LU tasks to speed up the WCSM test. The capability of the proposed algorithm was evaluated and compared with related work [2, 5], which demonstrated that our algorithm produced savings of 26–33% in the number of RMA iterations.

In future work we will extend the algorithm to schedulability tests for multiframe task sets [7], where the computation requirements of tasks in consecutive periods are modeled as regular patterns. For analyzing the schedulability of tasks with synchronization requirements, we shall also take the blocking time produced by priority inversion into consideration in WCSM tests.

6. REFERENCES


