Period-Dependent Initial Values for Exact Schedulability Test of Rate Monotonic Systems

Wan-Chen Lu[1], Kwei-Jay Lin[2], Hsin-Wen Wei[1], and Wei-Kuan Shih[1]

[1]National Tsing Hua University
Dept. of Computer Science
Hsinchu, Taiwan
{wanchen, bertha, wshih}
@rtlab.cs.nthu.edu.tw

[2]University of California, Irvine
Dept. of Electrical Engineering and
Computer Science, CA
klin@uci.edu

Abstract

Real-time systems using Rate Monotonic fixed priority scheduling can be checked for schedulability either by pessimistic schedulability conditions or exact testing. Exact testing provides a more precise result but cannot always be performed in polynomial time. Audsley et al. proposed one of the earliest methods by iteratively deriving the job response times. Other researchers have improved the efficiency of their exact test method by using different initial values. All currently proposed initial values do not use the relationship between task periods. In this paper we define initial values using the largest and the second largest periods in a system. We show that the new initial values can significantly improve the exact test.

1. Introduction

The schedulability of real-time periodic tasks using the Rate Monotonic (RM) fixed priority scheduling algorithm can be checked using the total utilization factor of all tasks in a system [6, 8, 12]. For a periodic task set, each periodic task τ_i is defined with two parameters (c_i, p_i), where c_i and p_i are the worst case computation time and the period of task τ_i respectively. The *utilization* factor of task τ_i is defined by $u_i = c_i/p_i$. If the total utilization of a system is less than or equal to a bound, the system is guaranteed to be schedulable.

The utilization bound provides a pessimistic testing since tasks may be schedulable even if they do not meet the bound condition. Lehoczky *et al*. [7] is among the first to propose the exact-test concept for rate monotonic analysis (RMA). In the RMA process, one needs to check if the total computation time needed by a task set before a time instance (after all tasks are ready) can be completed before that time. When the time demand at any time t ($\leq p_i$) is equal to t, we call this type of time point as *time demand schedulable (TDS) point* for task τ_i in this paper.

Audsley *et al.* [1] proposes another idea for RMA, in which the worst-case response time WR_i of task τ_i is derived by iteratively calculating the following formula:

$$WR^{(l+1)} = c_i + \sum_{j=1}^{i-1} \left\lceil WR^{(l)}/p_j \right\rceil c_j$$

until $WR^{(l+1)}$ either converges to a constant number (i.e., WR_i) or is beyond the deadline of task p_i, where $WR^{(0)} = WR_{i-1}+c_i$. In this paper, this RMA method is referred to as the *response time analysis* (RTA).

Although exact tests provide a better schedulability testing for RM tasks, they require the task response time to be calculated iteratively [1, 2, 3, 7, 11]. In recent years, researchers have proposed several methods to improve the run time of RMA tests [2, 3, 11]. Bini and Buttazzo [2] proposed a way to balance the required run time of the time demand analysis method and the false-identification rate of schedulable tasks. Bril *et al.* [3] proposed a new initial value $WR^{(0)} = \boldsymbol{max}\{WR_{i-1}+c_i, c_i/(1-(u_1+u_2+\ldots+u_{i-1}))\}$ used by RTA. More recently, Lu *et al.* [11] proposed an RTA iterative formula to derive $WR^{(l+1)}$ for $l \geq 0$.

This paper studies the initial values used in RTA and proposes new initial values with a much better performance. Previous works [1, 3] try to find the first TDS point in the critical interval of τ_i (see Def. 4). However, using our proposed initial values, the TDS point identified is not necessarily the first TDS, but maybe the kth TDS point (k is greater than 1) in the interval. We show in this paper that the task set is schedulable as long as some TDS point exists in the critical interval of τ_i. We propose the initial value as $\boldsymbol{max}\{p_i-p_{i-1}, p_i/2, c_i/(1-(u_1+u_2+\ldots+u_{i-1}))\}$. The performance of the proposed initial value has been tested and compared to earlier works [1, 3].

The result shows that our RTA method may achieve a saving of up to 78.2% in the number of iterations. When the number of tasks in a system is large, our method can significantly reduce the number of iterations.

The remainder of this paper is organized as follows. In Section 2, we define the notions for real-time periodic tasks. Section 3 presents some new initial values for the RTA schedulability test, an extended testing algorithm and its correctness. Section 4 shows the simulation results for the proposed initial values used for RTA. The paper is concluded in Section 5 with a comparison of all initial values proposed so far.

2. Definitions and Motivation

Before we show our new result, we first present some formal definitions about RM scheduling.

Definition 1. [8] Let **T** = $\{\tau_1, \tau_2, \ldots, \tau_i\}$ be a set of i periodic tasks. Each task τ_j ($j = 1, \ldots, i$) is a tuple (c_j, p_j), where c_j and p_j are the maximum computation time and the period of task τ_j, respectively. The utilization of τ_j, denoted by u_j, is equal to c_j/p_j. The total utilization of $\tau_1, \tau_2, \ldots, \tau_i$ is denoted as U_i. Without loss of generality, we may assume that tasks in **T** are indexed in the order of increasing periods, and hence task τ_i has the longest period of all tasks.

Definition 2. [9] *Feasible and Schedulable*: Suppose a task set is scheduled by ***S*** scheduling algorithm, the schedule for the task set is feasible if all the jobs in each task can meet their corresponding deadlines under ***S*** scheduling. A task set is schedulable if there exists a feasible schedule.

Definition 3. [9] *Rate Monotonic Scheduling*: Rate monotonic (RM) algorithm is an optimal fixed-priority scheduling algorithm, which means RM algorithm can always generate a feasible schedule if a task set is schedulable. Under RM scheduling, the individual task priorities are assigned inversely proportional to their respective periods. In other words, a task with shorter period is assigned with a higher priority (and can be scheduled earlier).

Definition 4. [8] *The Critical Instant and Critical Interval*: The critical instant of a periodic task is the beginning of the period when its computation time is requested simultaneously with the computation times of all higher priority tasks. The critical interval of a periodic task is the time interval between a critical instant and the deadline of the task.

Definition 5. *Time Demand and Time Demand Schedulable (TDS) Point*: For any time instant t ($\leq p_i$), the time demand at t, denoted by *TD(t)*, is equal to

$$c_i + \sum_{j=1}^{i-1}\left\lceil t/p_j \right\rceil c_j \,.$$

If *TD(t)* is *equal* to t, τ_i is RM schedulable and the time t is called the *time demand schedulable (TDS) point*.

Definition 6. [1] *Response Time Analysis (RTA)*: According to the exact test method proposed in [8], the worst-case response time WR_i of each task τ_i is derived by iteratively calculating the formula

$$WR^{(l+1)} = c_i + \sum_{j=1}^{i-1}\left\lceil WR^{(l)}/p_j \right\rceil c_j$$

unitl $WR^{(l+1)}$ either converges to a real number (i.e., WR_i) or exceeds the deadline of task τ_i. A task is schedulable if its worst-case response time, i.e. WR_i, is no larger than its deadline; otherwise, the task is unschedulable. Please notice that (1) $l \geq 0$ and (2) $WR^{(0)}$ is the initial value of the RTA iterative procedure and is equal to $WR_{i-1}+c_i$.

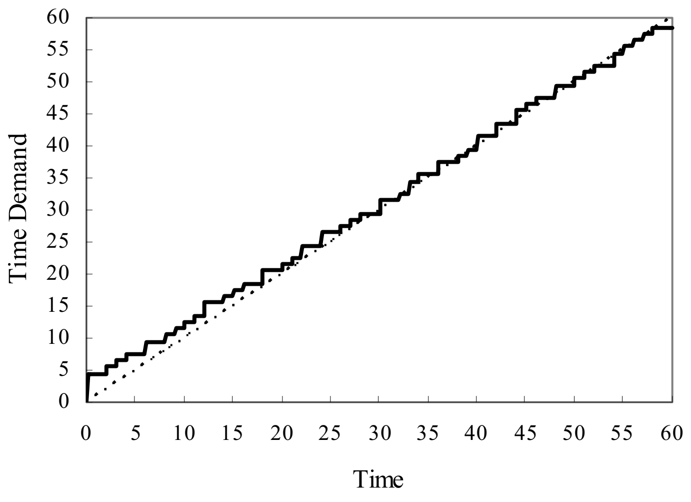
3. The Derivation of the Initial Value

Bril *et al.* [3] proposed an initial value $WR^{(0)}$ = ***max***$\{WR_{i-1}+c_i, c_i/(1-U_{i-1})\}$ for RTA. However, All RTA's with initial values proposed in [1, 3] are to find the first TDS point in the critical interval of τ_i to decide the schedulability. When the value of c_i is too small or U_{i-1} (i.e., the total utilization of tasks $\tau_1, \tau_2, \ldots, \tau_{i-1}$) is not large enough, $c_i/(1-U_{i-1})$ will be small and cannot produce a good initial value. For example, assume **T** = $\{\tau_1, \tau_2, \ldots, \tau_5\}$ is a set of 5 periodic tasks as shown in Table 1. The subset of **T** = $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ is known to be RM schedulable. If the schedulability of τ_5 is tested by RTA, $c_i/(1-U_{i-1})$ = 0.5/(1-0.94) = 9.85 would not be chosen as the initial value because $WR_{i-1}+c_i$ = 18+0.5 = 18.5 > 9.85.

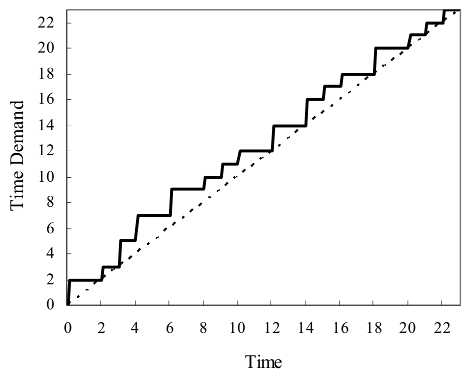
Table 1. The periodic task set T

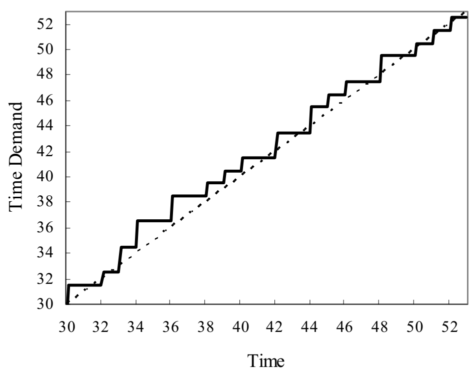
τ_i	c_i	p_i
τ_1	1	2
τ_2	1	3
τ_3	1	11
τ_4	1	40
τ_5	0.5	60

In this paper, new initial values are proposed by considering the largest and the second largest periods in a system. Our new initial values are motivated by the observation that many TDS points may exist in the critical interval of τ_i if τ_i is schedulable. Fig. 1(a) shows the time demand function for tasks of Table 1. When the time demand (solid line) intersects the time supply (45° dashed line), the intersected point is a TDS point. Therefore, there are 19 TDS points for the task set of Table 1. Earlier RTA methods [1, 3] try to find the first TDS point (i.e. 29.5) in the critical interval of τ_i. However, if the initial value is large enough (that is, larger than the first TDS point), other TDS points may be found in a smaller number of RTA iterations.

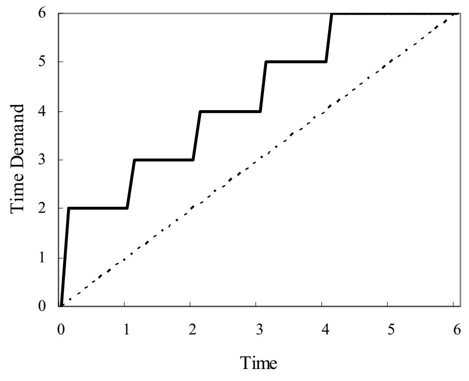


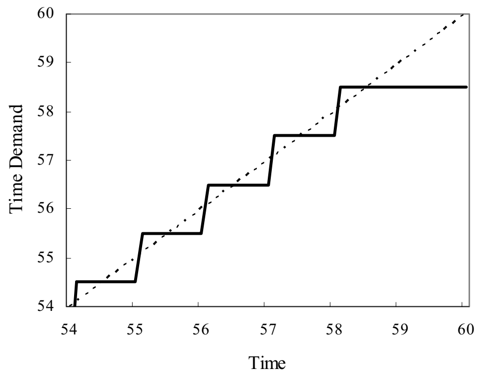
(a) The time demand relationship for **T** under different time instants





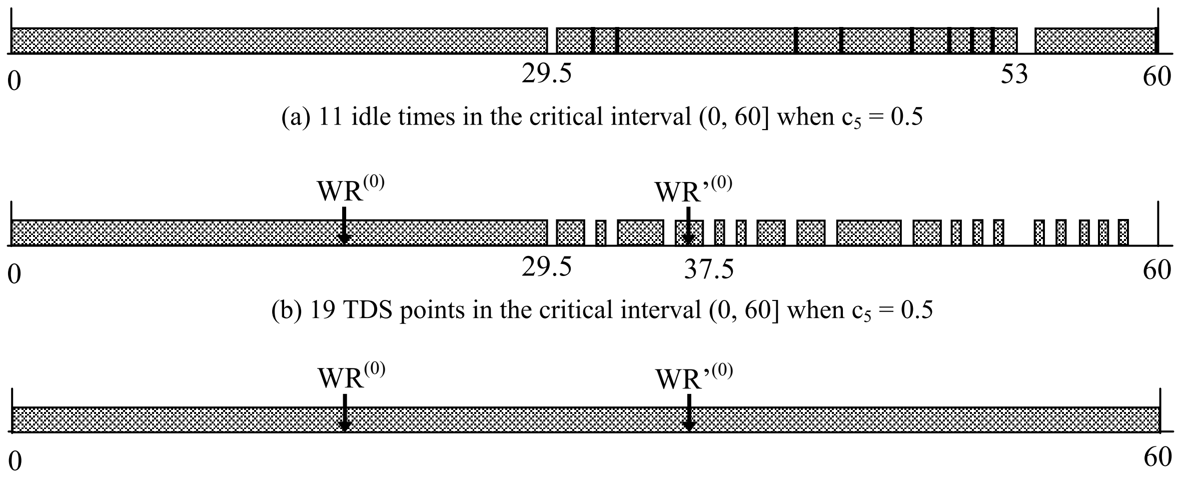
(b) The time demand relationship for **T_1'** in (0, 23] (c) The time demand relationship for **T** in (30, 53]





(d) The time demand relationship for **T_2'** in (0, 6] (e) The time demand relationship for **T** in (54, 60]

Fig. 1 The time demand relationship



(a) 11 idle times in the critical interval (0, 60] when $c_5 = 0.5$

(b) 19 TDS points in the critical interval (0, 60] when $c_5 = 0.5$

(c) No idle time and TDS point in the critical interval (0, 60] when $c_5 = 3$

Fig. 2. The initial value setting for RTA

Fig. 2(a) shows the actual RM schedule for tasks listed in Table 1. There are 11 idle times in the critical interval of τ_5 (i.e. (29.5, 30], (32, 32], (33, 33], (42, 42], (44, 44], (48, 48], (50, 50], (51, 51], (52, 52], (53, 54] and (60, 60]). (Please notice that only the first and tenth idle times are with length larger than 0.) However, there are 19 TDS points in the critical interval of τ_5, as shown in Fig. 1(a) and Fig. 2(b). In other words, an idle time existed in the critical interval of τ_i may generate at least one, but usually much more, TDS point. It may be faster to find a TDS point than to find an idle time interval.

We first study the relationship between the first idle time (with length 0.5) and TDS points in (30, 53]. In the time interval (30, 53], only $\tau_1, \tau_2, \ldots, \tau_4$ are scheduled. Fig. 1(b) shows the time demand function for tasks $\tau_1, \tau_2, \ldots, \tau_4$ (τ_5 is excluded) with the ready times of $\tau_1, \tau_2, \ldots, \tau_4$ are 0, 0, 3, 10 respectively (since after the first idle time, the first actual job release time of $\tau_1, \tau_2, \ldots, \tau_4$ are 30, 30, 33, 40 respectively). This task set is called **T₁'**. In Fig. 1(b), 9 TDS points (i.e. at time instants 2, 3, 12, 14, 18, 20, 21, 22, 23) appear before or at $t = 23$. However, in the critical interval of τ_5, an idle time with length 0.5 before $t = 30$ exists. Therefore, by deducting 0.5 from the time demand of each time instant shown in Fig. 1(b), the time demand function for the time interval (30, 53] in the critical interval of τ_5 could be obtained. Fig. 1(c) shows the time demand relationship for the time interval (30, 53] in the critical interval of τ_5. From Fig. 1(c), we could see that, due to the first idle time, 13 TDS points are in (30, 53].

Next, we investigate the relationship between the tenth idle time (with length 1) and TDS points in (54, 60]. In the time interval (54, 60], only $\tau_1, \tau_2, \ldots, \tau_4$ are scheduled. Fig. 1(d) shows the time demand relationship for tasks $\tau_1, \tau_2, \ldots, \tau_4$ (τ_5 is excluded) with the ready times of $\tau_1, \tau_2, \ldots, \tau_4$ are 0, 0, 1, 26 respectively (since after the second idle time, the first actual job release time of $\tau_1, \tau_2, \ldots, \tau_4$ are 54, 54, 55, 80 respectively). We call this task set **T₂'**. In Fig. 1(d), only one TDS point appears at $t = 6$. However, in the critical interval of τ_5, 10 idle times with length 1.5 before $t = 54$ exist. Therefore, by deducting 1.5 from the time demand of each time instant shown in Fig. 1(d), the time demand relationship for the time interval (54, 60] in the critical interval of τ_5 could be obtained. Fig. 1(e) shows the time demand function for the time interval (54, 60] in the critical interval of τ_5. From Fig. 1(e), we could see that 5 TDS points are in (54, 60].

From the above observation, we can see that, assume the kth idle time interval = $(X_k, Y_k]$ for $k \geq 1$, a TDS point t ranging between the kth and (k+1)th idle times can be generated if the following equation is met:

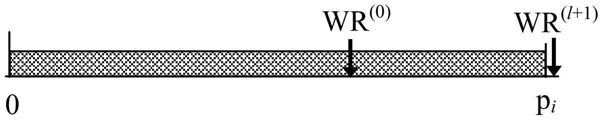
$$t = TD'(t) + Y_k - \sum_{m=1}^{k}(Y_m - X_m)$$

where $TD'(t) = \sum_{j=1}^{i-1}(\lceil t/p_j \rceil - \lceil Y_k/p_j \rceil)c_j$. When k is larger, so is $\sum_{m=1}^{k}(Y_m - X_m)$, and more TDS points could be found.

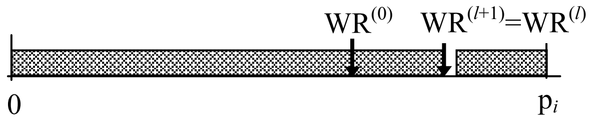
In Fig. 2(b), the gray areas (respectively, white areas) are a set of time instants at which the time demands are larger than (respectively, less than) the corresponding time instants, and the margin of the end of the kth gray area and the beginning of the kth white area (for $k \geq 1$) are TDS points. Hence, if the initial value belongs to the kth ($1 \leq k \leq 19$) gray area, the kth TDS point could be found by more than one RTA iterations; if the initial value belongs to the kth ($1 \leq k \leq 19$) white area or the margin of the kth gray and white areas (i.e. TDS point), the kth TDS point could be found in 1 RTA iteration. Therefore, when the value of $WR^{(0)} = \boldsymbol{max}\{WR_4+c_5, c_5/(1-U_4)\} = 18.5$, the first

TDS point, i.e. 29.5, would be found in 9 RTA iterations. If $WR^{(0)}$ is set to a larger value such as $WR'^{(0)}$, the fifth TDS point (i.e. 37.5) could be found only in 2 RTA iterations. Besides, if the computation time of τ_5 changes from 0.5 to 3, as shown in Fig. 2(c), τ_5 is not schedulable. Under this condition, the number of RTA iterations needed to check the schedulability of τ_5 must be less if $WR'^{(0)}$ is applied (rather than $WR^{(0)}$).

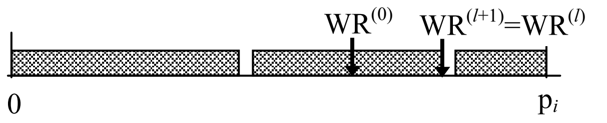
Therefore, we propose a new initial value to reduce the number of RTA iterations and prove that if the task set is schedulable, at least one TDS point can be found between this initial value and p_i. The proofs are shown in the following lemmas and theorems.



(a) **T** is not schedulable



(b) **T** is schedulable and the first idle time appearing after $WR^{(0)}$



(c) **T** is schedulable and the first idle time appearing before or at $WR^{(0)}$

Fig. 3. RTA when $WR^{(0)}$ = *max*{p_i-p_{i-1}, p_i/2}

Lemma 1. In the critical interval of τ_i if there is an idle time in the time interval (0, p_i-p_{i-1}], there must exist another idle time in the time interval (p_i-p_{i-1}, p_i].
Proof. If the first job of τ_i is scheduled before or at the time instant p_i-p_{i-1} by c_i units of time, only jobs of $\tau_1, \tau_2, \dots, \tau_{i-1}$ would be scheduled after p_i-p_{i-1}. Because $\tau_1, \tau_2, \dots, \tau_{i-1}$ are RM schedulable, at least one idle time must appear in the critical interval of τ_{i-1} (with length p_{i-1} and starting from the even phase, that is, the worse condition). At time instant p_i-p_{i-1}, jobs of $\tau_1, \tau_2, \dots, \tau_{i-1}$ may be released either in the even phase or in the odd phase, and therefore, there must also exist an idle time in the time interval (p_i-p_{i-1}, p_i]. ■

Lemma 2. In the critical interval of τ_i if there is an idle time in the time interval (0, p_i/2], there must exist another idle time in the time interval (p_i/2, p_i].
Proof. If there is an idle time in the first half of the critical interval of τ_i, there must also exist an idle time in the second half of the critical interval of τ_i. This is because at the critical instant of τ_i, jobs of $\tau_1, \tau_2, \dots, \tau_i$ are released at the same time (in the even phase, or the worse condition), but jobs of $\tau_1, \tau_2, \dots, \tau_i$ are released either in the even phase or in the odd phase at time instant p_i/2. Hence if there is an idle time in (0, p_i/2], another idle time must appear in (p_i/2, p_i]. ■

Theorem 1. In the critical interval of τ_i if there is an idle time in the time interval (0, ***max***{p_i-p_{i-1}, p_i/2}], there must exist another idle time in the time interval (***max***{p_i-p_{i-1}, p_i/2}, p_i].
Proof. If ***max***{p_i-p_{i-1}, p_i/2} = p_i-p_{i-1}, according to Lemma 1, we know that if there is an idle time in the time interval (0, p_i-p_{i-1}], there is another idle time in (p_i-p_{i-1}, p_i], that is, in (***max***{p_i-p_{i-1}, p_i/2}, p_i]. Otherwise, if ***max***{p_i-p_{i-1}, p_i/2} = p_i/2, according to Lemma 2, we can know that if there is an idle time in the time interval (0, p_i/2], there must be another idle time in (p_i/2,p_i], that is, in (***max***{p_i-p_{i-1},p_i/2}, p_i]. The theorem is therefore proved. ■

Theorem 2. The RTA schedulability test result for a periodic tasks set **T** is correct if $WR^{(0)}$ = ***max***{p_i-p_{i-1}, p_i/2}.
Proof. This theorem is proved by considering the following three cases:

Case 1: If τ_i is not schedulable, no idle times and TDS points could be found in the critical interval of τ_i, so the RTA iterative procedure would be done until $WR^{(l+1)} > p_i$ (for some $l \geq 0$), as shown in Fig. 3(a).

Case 2. If τ_i is schedulable and the first idle time appears after $WR^{(0)}$ in the critical interval of τ_i, the RTA iterative procedure would be done until $WR^{(l+1)} = WR^{(l)} \leq p_i$ (for some $l \geq 1$), as shown in Fig. 3(b).

Case 3. If τ_i is schedulable and the first idle time appears before or at $WR^{(0)}$ in the critical interval of τ_i, according to Theorem 1, there would be another idle time in ($WR^{(0)}$, p_i]. That is, at least one TDS point is in ($WR^{(0)}$, p_i]. Besides, if $WR^{(1)}$ is less than $WR^{(0)}$, one TDS point appears exactly at $WR^{(1)}$. Hence the RTA iterative procedure would be done until $WR^{(l+1)} = WR^{(l)} \leq p_i$ (for some $l \geq 0$) or $WR^{(1)} < WR^{(0)}$, as shown in Fig. 3(c). ■

We now extend the result in Theorem 2 by considering the initial values proposed in [1, 3], as shown in Corollary 1. Note that WR_{i-1} is the first idle time (i.e. the first TDS point) defined in [1, 3] and must be known from the schedulability test of τ_{i-1}. However, the TDS point found by our initial value may not be is the first one, and therefore, WR_{i-1}+c_i cannot be considered to be our initial value.

Corollary 1. The RTA schedulability test result for a periodic tasks set **T** is correct if $WR^{(0)} = \boldsymbol{max}\{p_i\text{-}p_{i\text{-}1}, p_i/2, c_i/(1\text{-}U_{i\text{-}1})\}$.

Under this initial value $\boldsymbol{max}\{p_i\text{-}p_{i\text{-}1}, p_i/2, c_i/(1\text{-}U_{i\text{-}1})\}$, a TDS point may exist exactly at time instant $WR^{(1)}$ which is less than $WR^{(0)}$. So the termination condition for the RTA iterative procedure has an extra condition: $WR^{(0)} > WR^{(1)}$. That is, if $WR^{(0)} > WR^{(1)}$, the task set must be schedulable. The modified RTA exact test algorithm is shown in Algorithm 1.

Algorithm 1: Exact Schedulability Test

Input: A periodic task set $\mathbf{T} = \{\tau_1, \tau_2, \ldots, \tau_i\}$ on condition that the subset of $\mathbf{T} = \{\tau_1, \tau_2, \ldots, \tau_{i\text{-}1}\}$ is schedulable;
Output: The schedulability result of **T**;

```
1  $WR^{(0)} = \boldsymbol{max}\{p_i\text{-}p_{i\text{-}1}, p_i/2, c_i/(1\text{-}U_{i\text{-}1})\}$;
2  $l = \text{-}1$;
3  do
4        $l = l+1$;
5        $WR^{(l+1)} = c_i + \sum_{k=1}^{i-1}\left\lceil WR^{(l)}/p_k \right\rceil c_k$ ;
6  while $WR^{(l+1)} > WR^{(l)}$ and $WR^{(l+1)} \le p_i$
7  if $WR^{(l+1)} \le WR^{(l)}$ then
8        return "schedulable";
9  else
10       return "unschedulable";
11 end if
```

The following two examples show the operations of Algorithm 1.

Example 1. Let $\mathbf{T} = \{\tau_1 = (1, 2), \tau_2 = (1, 3), \tau_3 = (1, 11), \tau_4 = (1, 40), \tau_5 = (0.5, 60)\}$ be a set of periodic tasks as shown in Table 1. According to Algorithm 1, the schedulability of τ_5 could be tested as follows: Initially, $WR^{(0)} = \boldsymbol{max}\{60\text{-}40, 60/2, 0.5/(1\text{-}0.94)\} = \boldsymbol{max}\ \{20, 30, 9.85\} = 30$. Then, $WR^{(1)} = 29.5 < WR^{(0)}$. Therefore, the termination condition is satisfied in 1 RTA iteration and the task set **T** is reported as "schedulable". From the pervious example, we could know that if the value of $WR^{(0)}$ is set based on the method in [3], that is, $WR^{(0)} = \boldsymbol{max}\{WR_4+c_5, c_5/(1\text{-}U_4)\} = 18.5$, the RTA iterative procedures would be done in 9 iterations ($WR^{(1)} = 20.5$, $WR^{(2)} = 21.5$, $WR^{(3)} = 22.5$, $WR^{(4)} = 24.5$, $WR^{(5)} = 26.5$, $WR^{(6)} = 27.5$, $WR^{(7)} = 28.5$, $WR^{(8)} = WR^{(9)} = 29.5$).

Example 2. Let $\mathbf{T} = \{\tau_1 = (1, 2), \tau_2 = (1, 3), \tau_3 = (1, 20), \tau_4 = (1.1, 33)\}$ be a set of periodic tasks. The schedulability of τ_4 is tested as follows: Initially, $WR^{(0)} = \boldsymbol{max}\{33\text{-}20, 33/2, 1.1/(1\text{-}0.88)\} = 16.5$. Then, $WR^{(1)} = WR^{(2)} = 17.1 \le p_4$ (i.e. 33). Therefore, the termination condition is met in 2 RTA iterations and **T** is "schedulable". If the value of $WR^{(0)}$ is set as $\boldsymbol{max}\{WR_3+c_4, c_4/(1\text{-}U_3)\} = \boldsymbol{max}\{6+1.1, 1.1/(1\text{-}0.88)\} = 9.42$, the RTA iterative procedure will be done in 7 iterations ($WR^{(1)} = 11.1$, $WR^{(2)} = 12.1$, $WR^{(3)} = 14.1$, $WR^{(4)} = 15.1$, $WR^{(5)} = 16.1$, $WR^{(6)} = WR^{(7)} = 17.1$). Note that if the period of τ_4 is changed from 33 to 40, Algorithm 1 will find the second TDS point (i.e. 19.1) in 1 RTA iteration and report "schedulable".

For tasks whose deadlines are no at the end of their periods, RTA can be performed similarly. Suppose a periodic task $\tau_i = (c_i, p_i, d_i)$ where d_i is the relative deadline of τ_i and $d_i \le p_i$. The schedulability test for τ_i could be done by modifying the following steps of Algorithm 1: (1) Step 1 of Algorithm 1 is changed to $WR^{(0)} = \boldsymbol{max}\{d_i\text{-}d_{i\text{-}1}, d_i/2, c_i/(1\text{-}U_{i\text{-}1})\}$; (2) Step 6 is changed to $WR^{(l+1)} > WR^{(l)}$ and $WR^{(l+1)} \le d_i$. With these changes, the TDS point is checked against the deadline.

4. Performance Evaluation

4.1. Data Set and Measurement

The primary performance metric reported is the *Iteration Ratio*. Let x and y be the numbers of iterations executed in RTA with initial value = $\boldsymbol{max}\{WR_{i\text{-}1}+c_i\}$ and one of the initial values (1) $\boldsymbol{max}\{WR_{i\text{-}1}+c_i, c_i/(1\text{-}U_{i\text{-}1})\}$ or (2) $\boldsymbol{max}\{p_i\text{-}p_{i\text{-}1}, p_i/2\}$ and (3) $\boldsymbol{max}\{p_i\text{-}p_{i\text{-}1}, p_i/2, c_i/(1\text{-}U_{i\text{-}1})\}$ respectively. The *Iteration Ratio* of RTA with a new initial value is defined as y/x.

The task sets for performance study are generated based on the benchmark systems used in [4, 5, 10, 13]. A random number generator is used to generate task sets: the number of tasks per task set, denoted by n, was randomly selected in (3, 5), (5, 10), (10, 15), and (15, 20) respectively. The number of fundamental period frequencies is a real number within the range [1/4, 1] multiplied by the number of tasks in the task set. The data sets are generated with a total utilization factor between 0.75 and 1. The experiments were started with a task set with a total utilization of 0.75, and repeated for sets with a total utilization factor increased by the increment of 0.05 until it reaches 1. The utilization factor of every task is no more than 0.4 of the total utilization factor of its task set. Every task is assigned a fundamental frequency randomly, with the possibility of assigning k fundamental frequencies equaling to $(1/2)^{k\text{-}1}$. The period of each task is the product of all its assigned fundamental frequencies. A total of 10,000 task sets were tested for each utilization factor.

4.2. Experimental Results for the Periodic Task Model

Fig. 4 shows the iteration ratios of the RTA's using initial values of (1) $\boldsymbol{max}\{WR_{i-1}+c_i,\ c_i/(1-U_{i-1})\}$, (2) $\boldsymbol{max}\{p_i-p_{i-1},\ p_i/2\}$, and (3) $\boldsymbol{max}\{p_i-p_{i-1},\ p_i/2,\ c_i/(1-U_{i-1})\}$ respectively. The number of tasks in a task set ranges between 3 and 5. As shown in Fig. 4, the iteration ratios of (2) and (3) are lower than that of (1) in most cases. (1) outperforms (2) only when the CPU utilization is equal to 1. This is because when the value of c_i is large or the total utilization of tasks $\tau_1, \tau_2, \ldots, \tau_{i-1}$ is large enough, $c_i/(1-U_{i-1})$ (i.e. (1)) can derive a large initial value to reduce the number of RTA iterations. Therefore, (3) could improve (2) further, especially when the CPU utilization is high. However, when the CPU utilization is low, (2) and (3) are much better than (1). This is because when the utilization is low, task sets are more likely schedulable and more than one TDS points may exist in the critical interval of τ_i. If $\boldsymbol{max}\{p_i-p_{i-1},\ p_i/2\}$ is large enough, a TDS point could be found in a small number of RTA iterations.

Figures 5, 6 and 7 show the iteration ratios when n is in (5, 10), (10, 15), and (15, 20) respectively. It is obvious that when n is large, the performance of (1) is not good, but both (2) and (3) have better performances. This is because with the same CPU utilization, c_i is more likely to be smaller when n becomes larger. Therefore, a smaller initial value would be derived in (1) and a larger number of RTA iterations is needed. From Fig. 4-7, we could see that (3) saves more then 78.2% (when $n = (15, 20)$ and the CPU utilization = 0.95) on the number of RTA iterations.

5. Conclusions

In this paper, we have presented two new initial values for exact schedulability tests on rate monotonic scheduling of periodic tasks. Our initial values are based on the largest and the second largest periods in a system. By considering these two periods, periodic tasks can be checked efficiently when performing the RM schedulability tests. A comparison of different initial values for RTA is shown in Table 2. For systems with a larger number of tasks, our new initial values may reduce the test iterations by as much as 78%. We will continue to use this idea to look for other engineering approaches to improve the schedulability of real-time systems.

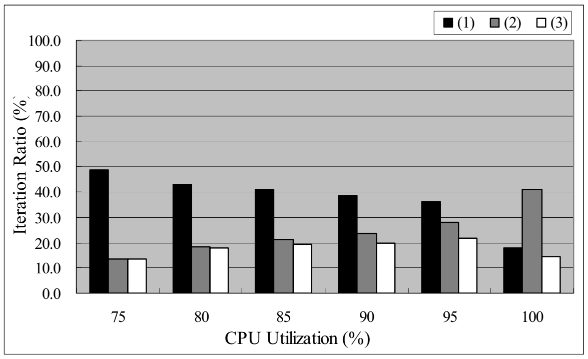


Fig. 4 Iteration ratio comparison when $n \in (3, 5)$

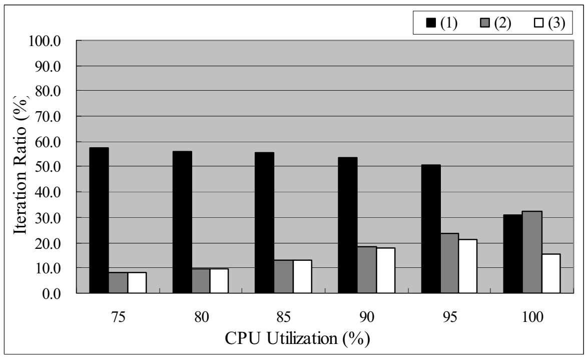


Fig. 5 Iteration ratio comparison when $n \in (5, 10)$

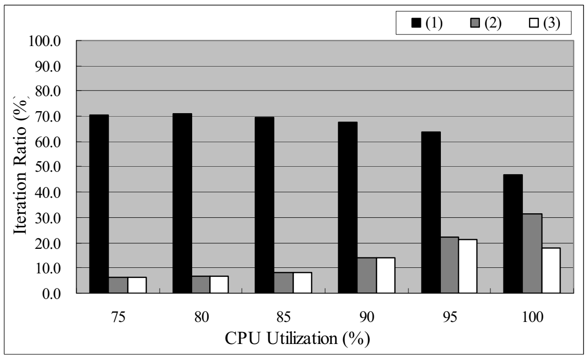


Fig. 6 Iteration ratio comparison when $n \in (10, 15)$

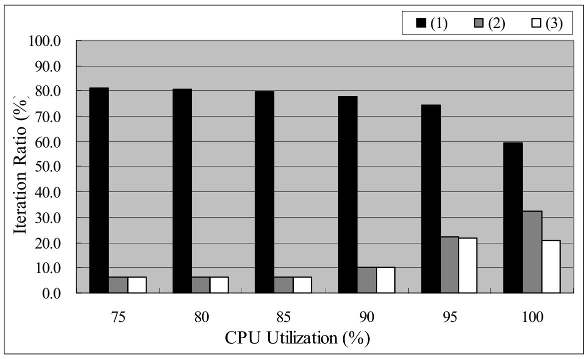


Fig. 7 Iteration ratio comparison when $n \in (15, 20)$

References

[1] N.C. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings, “Applying new scheduling theory to static priority pre-emptive scheduling,” *Software Engineering Journal*, 8(5), pp. 284-292, 1993.

[2] E. Bini and G.C. Buttazzo, “The Space of Rate Monotonic Schedulability,” *Proceedings of the 23rd IEEE Symposium on Real-Time Systems*, pp. 169-178, December 2002.

[3] R.J. Bril, W.F.J. Verhaegh, E.J.D. Pol, “Initial Values for On-line Response Time Calculations” *Proceedings of the 15th IEEE Euromicro Conference on Real-Time Systems*, pp. 13-22, July 2003.

[4] N.I. Kamenoff and N.H. Weiderman, “Hartstone Distributed Benchmark: Requirements and Definitions,” *Proc. of the 12th IEEE Real-Time Systems Symposium*, 1991.

[5] N. Kim, M. Ryu, S. Hong, M. Saksena, C.H. Choi, and H. Shin, “Visual Assessment of a Real-Time System Design: A Case Study on a CNC Controller,” *Proc. of the 17th IEEE Real-Time Systems Symposium*, 1996.

[6] S. Lauzac, R. Melhem, D. Mossè, “An Improved Rate-Monotonic Admission Control And Its Applications,” *IEEE Transactions on Computer*, 52(3), pp. 337-350, 2003.

[7] J.P. Lehoczky, L. Sha, and Y. Ding, “The Rate Monotonic Scheduling Algorithms: Exact Characterization and Average Case Behavior,” *Proceedings of the 10th IEEE Symposium on Real-Time Systems*, pp. 166-171, December 1989.

[8] C.L. Liu and J.W. Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment,” *Journal of the ACM*, 20(1), pp. 40-61, 1973.

[9] Jane W.S. Liu, *Real-Time Systems*, Prentice Hall, 2000.

[10] C.D. Locke, D.R. Vogel, and T.J. Mesler, “Building a Predictable Avionics Platform in Ada: A Case Study,” *Proc. of 12th IEEE Real-Time Systems Symposium*, 1991.

[11] W.C. Lu, J.W. Hsieh, W.K. Shih and T.W. Kuo, “A Faster Exact Schedulability Analysis for Fixed-priority Scheduling”, *Journal of Systems and Software*, 79(12), pp. 1744-1753, 2006.

[12] W.C. Lu, H.W. Wei, K.J. Lin, “Rate Monotonic Schedulability Conditions Using Relative Period Ratios,” *12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 3-9, 2006.

[13] J.J. Molini, S.K. Maimon, and P.H. Watson, “Real-Time System Scenarios,” *Proc. of the 12th IEEE Real-Time Systems Symposium*, 1991.

Table 2. The comparison of different initial values for RTA

$WR^{(0)}$	Strength	Weakness
$WR_{i-1}+c_i$	1. This is an intuitive initial value	1. Incremental computation is needed to derive WR_{i-1} 2. If the task is schedulable, the first TDS point is found in the critical interval 3. The performance is poor compared to RTA using other initial values
$c_i/(1-U_{i-1})$	1. No incremental computation is needed to derive the initial value 2. The performance is good when the number of tasks is small (≤ 5), see [3].	1. If the task is schedulable, the first TDS point is found in the critical interval 2. The performance is poor when the number of tasks is large (>5)
p_i-p_{i-1}	1. No incremental computation is needed to derive the initial value 2. If the task is schedulable, a TDS point which is not limited to the first one will be found in the critical interval	1. When the difference between p_{i-1} and p_i is small, the initial value may be too small
$p_i/2$	1. No incremental computation is needed to derive the initial value 2. If the task is schedulable, a TDS point which is not limited to the first one will be found in the critical interval	1. When the task is not schedulable, the initial value may be too small