

## A comparative study of rate monotonic schedulability tests

Nasro Min-Allah · Samee Ullah Khan ·  
Nasir Ghani · Juan Li · Lizhe Wang ·  
Pascal Bouvry

Published online: 2 February 2011  
© Springer Science+Business Media, LLC 2011

**Abstract** With the increased penetration of real-time systems into our surroundings, the selection of an efficient schedulability test under fixed priority system from a plethora of existing results, has become a matter of primary interest to real-time system designers. The need for a faster schedulability tests becomes more prominent when it applies to online systems, where processor time is a sacred resource and it is of central importance to assign processor to execute tasks instead of determining system schedulability. Under fixed priority nonpreemptive real-time systems, current schedulability tests (in exact form) can be divided into: response time based tests, and scheduling points tests. To the best of our knowledge, no comparative study of these test to date has ever been presented. The aim of this work is to assist the system designers in the process of selecting a suitable technique from the existing literature

---

N. Min-Allah  
COMSATS Institute of Information Technology, Islamabad 44000, Pakistan  
e-mail: [nasar@comstats.edu.pk](mailto:nasar@comstats.edu.pk)

S.U. Khan (✉) · J. Li  
North Dakota State University, Fargo, ND 58108-6050, USA  
e-mail: [samee.khan@ndsu.edu](mailto:samee.khan@ndsu.edu)

J. Li  
e-mail: [j.li@ndsu.edu](mailto:j.li@ndsu.edu)

N. Ghani  
University of New Mexico, Albuquerque, NM 87131-0001, USA  
e-mail: [nghani@ece.unm.edu](mailto:nghani@ece.unm.edu)

L. Wang  
Indiana University, Bloomington, IN 47408, USA  
e-mail: [wanglizh@indiana.edu](mailto:wanglizh@indiana.edu)

P. Bouvry  
University of Luxembourg, Luxembourg, Luxembourg  
e-mail: [pascal.bouvry@uni.lu](mailto:pascal.bouvry@uni.lu)

after knowing the pros and cons associated with these tests. We highlight the mechanism behind the feasibility tests, theoretically and experimentally. Our experimental results show that response time based tests are faster than scheduling points tests, which make the response time based tests an excellent choice for online systems.

**Keywords** Real-time systems · Feasibility analysis · Rate monotonic scheduling

## 1 Introduction

The main issue involved in the design of real-time systems is of timing correctness and the solution lies in the optimal scheduling of tasks on the given platform. These tasks are normally scheduled by a scheduling algorithms which assign priorities to task based on some predefined criteria such as activation rate or deadline, etc. The most commonly used approach to schedule real time tasks is priority driven which falls into two types: fixed priority and dynamic priority [6, 9]. A fixed-priority algorithm assigns the fixed priority to all jobs in each task, while dynamic-priority scheduling algorithms place no restrictions upon the manner in which priorities are assigned to individual jobs. Although dynamic algorithms are considered better theoretically, such techniques become unpredictable when transient overload occurs [2, 5, 11]. Because of its applicability, reliability, and simplicity, we only consider fixed-priority scheduling here.

In 1973, Liu and Layland started off by formalizing the real-time scheduling theory and ended up with a seminal paper [13], where it is shown that Rate Monotonic (RM) scheduling policy is optimal (when task deadlines and periods are the equal) under fixed priority, subject to the simplified timing constraints. According to [13], the task set is always feasible under rate monotonic scheduling scheme if the system utilization is below an upper bound called LL-bound. Consequently, in the last 35 years that have followed, many of these assumption are relaxed and the effect of such changes are highlighted on the rate monotonic scheduling policy. Eventually, a plethora of corresponding feasibility conditions are presented in literature [1, 2, 6, 8–10, 14, 16, 19].

Let  $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$  represent a nonconcrete periodic task system having periodic tasks. A nonconcrete periodic task  $\tau_i$  recurs and is represented by a tuple  $(c_i, d_i, p_i)$ , where  $c_i, d_i, p_i$  represent the computation time, relative deadline, and task period, respectively. In our model of a hard real-time task set, each task  $\tau_i$  generates a job  $j_i$  at each integer multiple of  $p_i$  and each such job has an execution requirement of  $c_i$  time units that must be completed by the next integer multiple of  $p_i$ . Moreover, all tasks immediately get ready for execution on uniprocessor as soon as they are released and all tasks overheads, such as task swapping times, etc. are subsumed into task computation times. Furthermore, we assume that initially, all of the tasks arrive simultaneously at  $t = 0$ .

RM assigns static priorities on task activation rates (periods) such that for any two tasks  $\tau_i$  and  $\tau_j$ ,  $\text{priority}(\tau_i) > \text{priority}(\tau_j) \Rightarrow \text{period}(\tau_i) < \text{period}(\tau_j)$ , while ties are broken arbitrarily. RM is optimal in the sense that if any task set can be scheduled with a fixed-priority assignment scheme, then the given task set also can

be scheduled with a rate-monotonic scheme. Due to its implicit characteristics such as simplicity and reliability, the RM has become the *de facto* standard supported by the USA Department of Defense and many other organization/manufacturers, such as IBM and General Motors [18].

For validating timing constrains under fixed priority systems, feasibility analyses—given a real-time application and processing resource, determining whether it is possible to meet all the deadlines under RM scheduling algorithm—are performed to achieve system predictability. To obtain a sufficient RM schedulability condition, authors in [13] made the following major assumptions for RM algorithm.

1. All tasks in the task set  $\Gamma$  are independent; there are no precedence nor resource constraints.
2. The task model is implicit deadline; relative deadline of a task is equal to its period.
3. All tasks in  $\Gamma$  are periodic; the instances of a periodic task are regularly activated at a constant rate.
4. Only task's computation demands are important.
5. Number of priority levels is unlimited; each task  $\tau_i$  has a unique priority  $\Omega_i$ .
6. All tasks in  $\Gamma$  are preemptive and the cost of preemption and other overheads are negligible.

In the rest of the paper, the task model refers to the above task model (implicit deadline model), which is well studied in literature. For each task  $\tau_i$ , its utilization is defined as:  $u_i = c_i/p_i$ . We define a cumulative CPU utilization  $U$  of periodic task system  $\Gamma$  as  $U = \sum_{i=1}^n u_i$ . The first feasibility test for RM was proposed in [13], called LL-bound: a periodic task system where  $d = p$  is static-priority feasible if

$$U \leq n(2^{1/n} - 1) \quad (1)$$

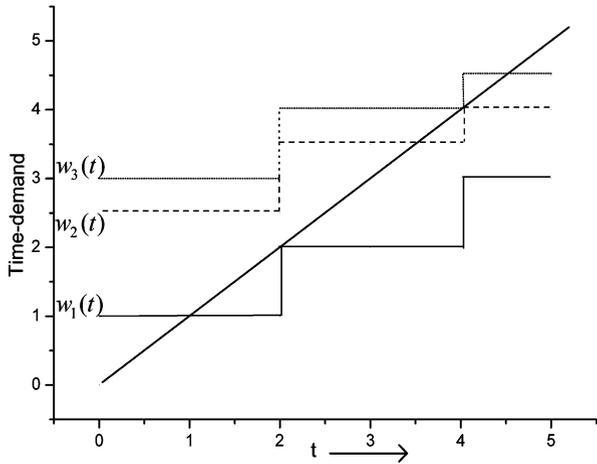
where  $n$  denotes the number of tasks in  $\Gamma$ . The term  $n(2^{1/n} - 1)$ , decreases monotonically from 0.83 when  $n = 2$  to  $\ln(2)$  as  $n \rightarrow \infty$ . This shows that any periodic task set of any size is static priority feasible upon a preemptive uniprocessor, if the RM scheduling is used and  $U$  is not greater than 0.693. This result gives a simple  $O(n)$  procedure to test system feasibility online, where tasks can arrive at run time; however, it is a sufficient condition only. It is quite possible that an implicit-deadline synchronous periodic task system which exceeds the LL-bound be static-priority feasible. A better utilization based test, called hyperbolic bound (H-bound) is proposed by Bini et al. in [3], which says that  $\Gamma$  is able to be scheduled if

$$\prod_{i=1}^n (u_i + 1) \leq 2 \quad (2)$$

Both (1) and (2) are sufficient conditions only; nothing can be concluded for RM schedulability of  $\Gamma$  when the above conditions are false. In Sect. 2, we briefly discuss some necessary and sufficient conditions from available literature [1, 2, 10, 14, 19].

The rest of this work is organized as follows. In Sect. 2, we present the exact feasibility tests in broader classes followed by a discussion in Sect. 3. Implementation and experimental results are discussed in Sect. 4, while Sect. 5 concludes the paper.

**Fig. 1** Time-demand analysis for  $\tau_1$ ,  $\tau_2$ , and  $\tau_3$



**2 Exact feasibility tests**

Rate Monotonic Analysis (RMA) is the classical way of determining RM task feasibility. Existing RMA techniques, which exhibit both the necessary and sufficient conditions can be further divided into two classes: scheduling points tests and response time based tests. A brief discussion on these tests is provided in the following.

**2.1 Scheduling points tests**

The workload constituted by  $\tau_i$  at time  $t$ , consists of its execution demand  $c_i$  as well as the interference it encounters due to higher priority tasks from  $\tau_{i-1}$  to  $\tau_1$ , and can be expressed mathematically as

$$w_i(t) = c_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil c_j \tag{3}$$

A periodic task  $\tau_i$  is feasible if we find some  $t \in [0, t]$  satisfying

$$L_i = \min_{0 < t \leq p_i} (w_i(t) \leq t) \tag{4}$$

In other words, task  $\tau_i$  completes its computation requirements at time  $t \in [0, t]$ , if and only if the entire request from the  $i - 1$  higher priority tasks and computation time of  $\tau_i$  is completed at or before time  $t$ . As  $t$  is a continuous variable, there are infinite numbers of points to be tested.

The feasibility of a task must be checked under worst case scenarios. The worst case combination for periodic tasks occurs at critical instant [13]. A critical instant for a periodic task  $\tau_i$  is the time when  $\tau_i$  is released simultaneously with request from all higher priority tasks.

2.1.1 Time demand analysis (TDA)

The first attempt to limit the infinite number of points in interval  $t \in [0, t]$  is made in [12]. The authors in [12] show that  $w_i(t)$  is constant, except at finite number of points, where tasks are released, called RM scheduling points. Consequently, to determine whether  $\tau_i$  is schedulable,  $w_i(t)$  is computed only at multiples of  $\tau_i \leq \tau_j, 1 \leq j \leq i$ . Specifically, let

$$S_i = \{ap_b | b = 1, \dots, i; a = 1, \dots, \lfloor p_i/p_b \rfloor\} \tag{5}$$

We have the following fundamental theorem to determine whether an individual task is feasible or not.

**Theorem 1** [12] *Given a set of  $n$  periodic tasks  $\tau_1, \dots, \tau_n, \tau_i$  can be feasibly scheduled for all tasks phasings using RM iff*

$$L_i = \min_{t \in S_i} \frac{w_i(t)}{t} \leq 1 \tag{6}$$

With above theorem,  $L_i$  is needed to be analyzed only at a finite number of points. The time demand  $w_i(t)$  for task  $\tau_i$  is tested at all scheduling point  $S_i$  and the point at which  $w_i(t)$  intersects the line with slope 1 is called feasible point, as illustrated in Fig. 1. When  $w_i(t)$  is above the thick line, it means the demand is higher than the available time  $t$ . The task  $\tau_i$  is declared schedulable according to the above feasibility condition (6) when the first feasible scheduling point  $t$  is encountered where cumulative demand is less than or equal to the corresponding task deadline  $d_i$ . The entire task set  $\Gamma$  is feasible iff

$$L = \max_{1 \leq i \leq n} \left\{ \min_{0 < t \leq p_i} \frac{w_i(t)}{t} \right\} \leq 1 \tag{7}$$

The time complexity of the above condition depends on both the number of tasks and maximum task period, i.e.,  $O(np_n/p_1)$ . The above mentioned feasibility test is subject to fixed-priority tasks whose response time is short, i.e.,  $d_i \leq p_i$ . In this paper, we only consider the case when  $d_i = p_i$ ; however, the case  $d_i \leq p_i$  can be easily addressed with above conditions. The time demand  $w_1(t), w_2(t), w_3(t)$  for three tasks  $\tau_1, \tau_2,$  and  $\tau_3,$  respectively, is drawn in Fig. 1, where the demand  $w_1(t)$  is satisfied first, followed by  $w_2(t)$  and lastly  $w_3(t)$  is satisfied, as per RM policy. In case of an unschedulable task  $\tau_i,$  the total demand  $w_i(t)$  never intersects the line with slop 1. It can be seen in Fig. 1 that for any task  $\tau_i,$  the set of points  $S_i$  consists of task periods and multiple of the task periods of all higher priority tasks  $\{\tau_1, \tau_2, \dots, \tau_{i-1}\}$  plus the period of the task  $\tau_i$  itself, i.e.,  $p_i$ . The feasibility test progresses at all points in ascending order and once the RM schedulability is confirmed the test stops, otherwise it has to check feasibility at all points till  $p_i$  and if no feasible points that can satisfy (6) is encountered, the task is declared unschedulable with RM.

For the task set whose relative deadlines are larger than their respective periods, there may be more than one active job at any time  $t$  and such situations are beyond the scope of this work. The pseudo code for TDA can be written as follows (Algorithm 1).

---

**Algorithm 1** Time demand analysis

---

```

procedure Boolean Time-Demand-Analysis( $\tau$ )
  for all  $\tau_i \in \tau$  do
    { Compute  $S_i = ap_b | b = 1, \dots, i; a = 1, \dots, \lfloor \frac{d_i}{p_j} \rfloor$ ;
    for all  $t \in S_i$  do
      if  $(L_i \leq 1)$  then
         $\tau_i$  is schedulable; break;
      end if
    end for
  end for
  if  $(L \leq 1)$  then
     $\tau$  is feasible;
  else  $\tau$  is infeasible;
  end if
end function

```

---

2.1.2 Hyper-planes exact test (HET)

To reduce the number of scheduling points, Bini and Buttazzo provided a formulation, called Hyper-planes exact test (HET) recently in [2], which reduces scheduling point for  $\tau_i$  from set  $S_i$  to a reduced set  $H_i$ . For any task  $\tau_i$ , HET begins with  $p_i$  and expands its search space by

$$H_i(t) = H_{i-1} \left( \left\lfloor \frac{t}{p_i} \right\rfloor p_i \right) \cup H_{i-1}(t) \tag{8}$$

where  $H_0(t) = \{t\}$ .

For more details on HET, the interested reader is referred to [2]. Below, we borrow the pseudocode for the HET from [2] and can be written as follows (Algorithm 2).

2.2 Response time analysis (RTA)

As discussed above, one method for avoiding all infinite points  $t$  in the range  $[0, p_i]$  is to test  $\tau_i$ 's schedulability at  $t \in S_i$  (for TDA) or  $t \in H_i$  (for HET), the other alternative is by iteration. To do so, the worst case workload of  $\tau_i$  within  $[0, p_i]$  can be expressed in the form

$$t = c_i + \sum_{j \in T} \left\lceil \frac{t}{p_j} \right\rceil c_j \tag{9}$$

To compute  $R_i$ , which is equal to the smallest value of  $t$  that satisfies the above equation, authors in [1, 19] proposed an iterative method to solve (9), called response time analysis (RTA). To find  $R_i$ , (9) can be solved with fixed-point iteration, starting from an initial guess  $R_i^0$ . Since the response time  $R_i$  of a task  $\tau_i$  is at least equal to its own execution time, so  $R_i^0 = c_i$ . Let  $R_i^{\#n}$  be the  $n$ -th approximation to the true value

**Algorithm 2** Hyper-planes exact test

```

procedure Boolean RMTTest( $\tau_n$ )
  int  $i$ ;
  for all ( $i = 0; i < n; i++$ ) do
    if ( $c_i + \text{WorkLoad}(i - 1, p_i) > p_i$ ) then
      return false;
    else
      return true;
    end if
  end for
end procedure
/*.....end of procedure.....*/
double lastPsi[BIG-ENOUGH]
double lastWorkLoad[BIG-ENOUGH]
function double WorkLoad(int  $i$ , double  $b$ )
  int  $f, g$ ; double branch0, branch1;
  if ( $i \leq 0$ ) then
    return 0;
  end if
  if ( $b \leq \text{last}\psi[i]$ ) then
    return lastWorkLoad[ $i$ ];
  end if
   $f = \lfloor \frac{b}{p_i} \rfloor$ ;  $g = \lceil \frac{b}{p_i} \rceil$ ;
  branch0 =  $b - f(p_i - c_i) + \text{WorkLoad}(i - 1, f \times p_{i-1})$ ;
  branch1 =  $g \times c_i + \text{WorkLoad}(i - 1, b)$ ;
  last $\psi[i] = b_i$ ;
  lastWorkLoad[ $i$ ] = min(branch0, branch1);
  return lastWorkLoad[ $i$ ];
end function

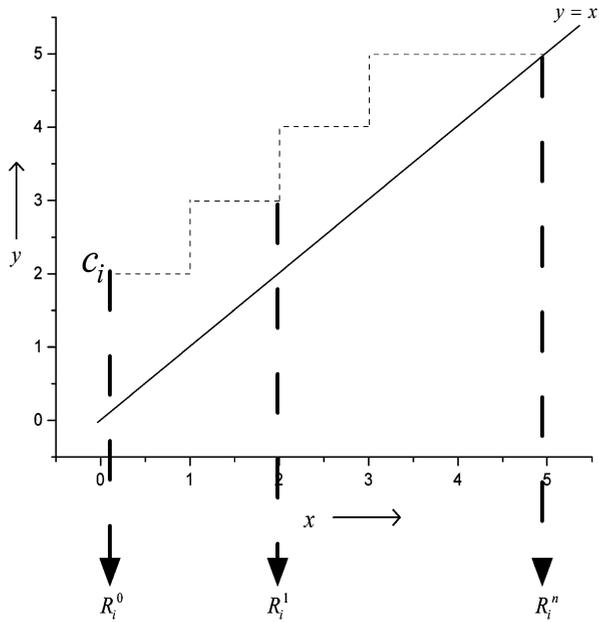
```

of  $R_i$ . During the  $l$ -th iteration for  $l \geq 1$ ,  $R_i^{\#l+1}$  can be computed by

$$R_i^{\#l+1} = c_i + \sum_{j \in T} \left\lceil \frac{R_i^{\#l}}{p_j} \right\rceil c_j \tag{10}$$

which converges after a finite number of iterations, as the sum is a monotonically increasing function of  $l$ . The loop is terminated either when  $R_i^{\#l+1} = R_i^{\#l}$  and  $R_i^{\#l} \leq d_i$  for some  $l$  or when  $R_i^{\#l+1} > d_i$ , whichever occurs first. In the former case  $\tau_i$  is schedulable ( $R_i \leq d_i$ ), while  $\tau_i$  is not schedulable in latter case. In Fig. 2, the line  $y = x$  makes an angle of  $45^\circ$  with  $x$  axis. The point where  $t$  intersects the line provides the maximum possible worst case response time of any job  $j_i \in \tau_i$ . When task deadlines are equal to period (implicit-deadline model), only one such jobs exist at any time  $t$ . To determine schedulability of a task, it is assumed that all jobs are released at  $t = 0$ , in such cases,  $j_{i,1} \in \tau_i$  has the longest response time, denoted by  $R_i$ . If  $R_i \leq d_i$ ,  $\tau_i$  is RM-feasible. An improvement to RTA is proposed in [19] called response time

**Fig. 2** Explanation of fixed-point iteration test for  $\tau_i$



improved (RTI) that utilizes  $R_{i-1}^0$  for assigning initial values to  $R_i$ , i.e.,  $R_i^0 = R_{i-1}^0 + c_i$  and thus converge faster than RTA.

Pseudocode for the RTA and RTI can be written as follows (Algorithms 3, 4).

### 3 Superiority of fixed-point techniques over scheduling points approaches

Techniques based on scheduling point methods (TDA/HET) have many advantages over iterative counterparts (RTA/RTI), such as being the fundamental ones and can be used at the system design time, while RTA is simply considered as a loop that can only answer feasibility problem [2, 4]. Scheduling points based tests are directly applicable to sensitivity analysis and help the system designers to fine tune their systems by maximizing some objective functions. On the other hand, due to its faster convergence time, iterative techniques have become the choice for current real-time systems in general and for online systems in particular.

In Fig. 3, we compare the two classical approaches discussed above (TDA and RTA). The graphical interpretation of the fixed-point (for RTA) and the scheduling point (for TDA) for a task  $\tau_i$  is shown with squares and black dots, respectively, in Fig. 3. The black dot shows a scheduling point (where  $w_i$  has to be tested at), which are potential candidates to be tested by TDA. Similarly, the squares shows the value of  $R_i$  at  $R_i^0, R_i^1, R_i^l$ , and  $R_i^n$ , respectively. The slope of the line is 1 and x-axis shows the available time while y-axis shows the cumulative demand for task  $\tau_i$ . Let the small rectangle denotes the point where feasibility of  $\tau_i$  is concluded, say the task  $\tau_i$  is schedulable by then. Both techniques, TDA and RTA are reaching the same point, but the way they hit the rectangle (intersects the line) is different. The larger

**Algorithm 3** Response time analysis

```

procedure bSchedulable RTA(tasks-vector);
double bSchedulable;
current-taskset-size = size(tasks-vector,1)
for all i=1:current-taskset-size do
    if demand(tasks-vector,i) > tasks-vector(i,2) then
        bSchedulable =false;
        return
    end if
end for
bSchedulable = true;
end procedure
/*.....end of procedure.....*/
function double WCRT(taskset,taskindex);
int r=0;
int rold =0;
current-taskset-size = size(taskset);
for all i = 1:taskindex do
    r=taskset(i,1);
end for
while r > rold;
    rold = r;
    r= taskset(taskindex,1);
for all i = 1:taskindex-1 do
        r = r + taskset(i,1) × ceil(rold/taskset(i,2));
end for
if r > taskset(taskindex,2) then
    break;
end if
end-while
end function

```

the distance between any two consecutive points the faster the technique is. From theoretical point of view, TDA is more fundamental to any task  $\tau_i$ , due to set  $S_i = lp_j$ ; it is known in advance at which point task feasibility will be tested by considering only task periods, i.e.,  $t \in \{(1, 2, \dots, \lfloor p_i/p_j \rfloor)p_j\}$  for  $1 \leq j \leq i$ . These are the points where workload actually increases monotonically. The number of points to be tested with TDA increases unreasonably when  $p_n \gg p_1$ . For instance when  $p_n = 10000$  and  $p_1 = 10$ , then roughly some 1,000 points need to be tested with TDA, for determining feasibility of the task set. Techniques such as sensitivity analysis [4, 7, 17] are directly applicable to scheduling points tests. In contrast, fixed-point techniques do not offer such facility, as  $R_i^l$  directly influence the expression  $c_i + \sum_{j \in T} \lceil R_i^{\#l}/p_j \rceil c_j$  for  $1 \leq j \leq i$  and unless both  $(c_i \text{ and } p_i)$  are known, next point  $R_i^{\#l+1}$  can not be determined. Figure 3 shows RTA gets larger jumps because it can skip many scheduling points due to its implicit nature which is a clear advantage over TDA. These larger jumps result

**Algorithm 4** Response time analysis improved

---

```

procedure bFeasible RTI(tasks-vector)
  int n ;
  n= size(tasks-vector,1);
  bFeasible = 1;
  int R = 0; Rold = 0;
  for all i=1:n do
    R = R+ tasks-vector(i,1);
    while (R > Rold)
      Rold = R;
      R = tasks-vector(i,1);
    for all k=1:i-1 do
      R = R + ceil(Rold/tasks-vector(k,2)) × tasks-vector(k,1);
    end for
    if (R > tasks-vector(i,2)) then
      bFeasible = 0;
      break;
    end if
  end for
  if (bFeasible == 0) then
    break;
  end if
end-while
end function

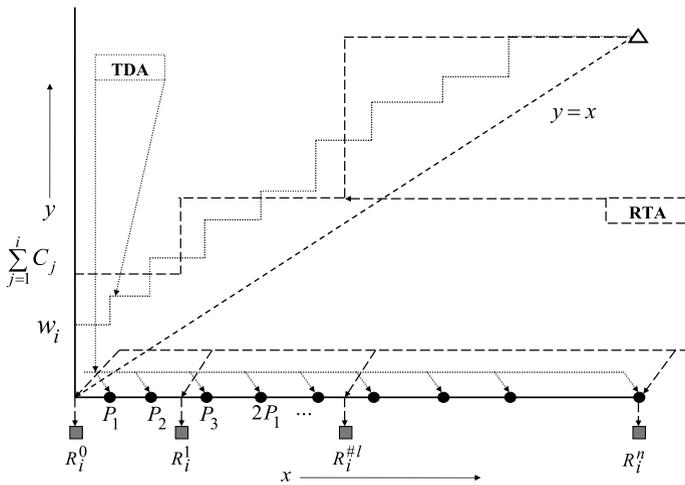
```

---

in converging much early and thus favors iterative techniques as efficient solutions for determining system feasibility online.

#### 4 Experimental results and analysis

In order to compare the necessary and sufficient conditions discussed above, we generate random task sets from size 5 to 35 with step size of 5. Task periods are also generated in the range of [10, 10000] with uniform distribution. Similarly, for corresponding task execution demands, random values are taken in the range of  $[1, p_i]$ , also with uniform distribution. Tasks priorities are assigned according to RM assignment algorithm, i.e., the smaller is the task period, the higher is the task priority. Figure 4 depicts the feasibility of task sets by using the scheduling points techniques and iterative tests. Utilization of the CPU is kept at 70%. The reason behind keeping this utilization is that it generates a task set that is always RM feasible. This arrangement results in testing feasibility of all the task sets by the respective feasibility tests. Experimentation is done on Pentium-VI (Intel, Core-2 CPU), 1.4 GHz with 2 GB RAM. From Fig. 4, it is clear when the tasks set has only 5 tasks, the time taken (in milliseconds) by response time analysis and response time improved is less than that taken by TDA/HET, noticeably better than scheduling point techniques. As the number of tasks increases, the time taken by all tasks tests also increases uniformly, which



**Fig. 3** Mechanisms adapted by SPT and RTA for determining next testing point

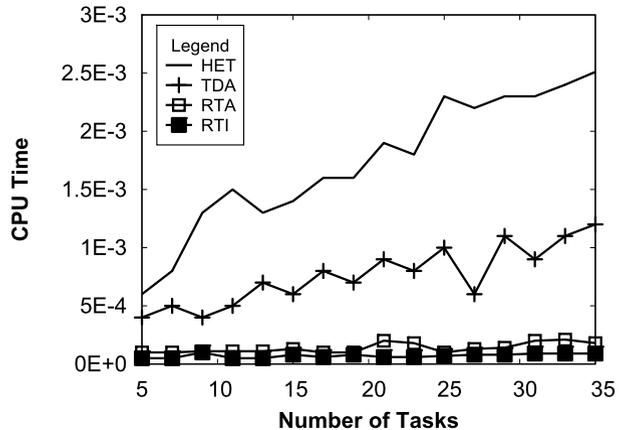
is understandable as more tasks are tested. The difference in performance between both classes of feasibility tests becomes more apparent when these tests are applied to larger task sets. As discussed in [7, 15], HET reduces number of scheduling points and number of inner most loops, however, its efficiency is compromised in the process and when CPU time is taken as performance evaluation metric, it exhibits poor performance. It can be seen that our experimental results are in accordance with theory, as discussed in earlier sections. In other words, scheduling points test (TDA/HET) are slower in performance as compared to iterative techniques (RTA/RTI). This gap in performance is due to testing of feasibility at all scheduling points for all tasks in the set with scheduling points tests, while iterative techniques have the advantage of making larger jumps in  $t$  that result in skipping a large number of scheduling points and hence feasibility of a task is determined much early.

### 5 Conclusion and future work

Existing rate-monotonic feasibility tests are discussed and a comparative analysis is made. The work is divided into two broader classes: scheduling points tests and response time bases tests. Both classes are discussed from theoretically, graphically and a comparison is made accordingly. It is concluded that being fundamental, scheduling points techniques are superior to response time based tests from design perspective. On the other hand, response time analysis tests outclass scheduling points tests as far as the analysis time is concerned.

As a future work, an interesting work could be the comparative study of feasibility tests with more relaxed assumptions and also a comparative study can be made to determine which test is more susceptible to dynamic voltage scaling (DVS) techniques.

**Fig. 4** Comparison of scheduling point tests and iterative techniques when CPU utilization = 0.70



## References

1. Audsley NC, Burns, A, Richardson, M, Wellings, A (1993) Applying new scheduling theory to static priority preemptive scheduling. *Softw Eng J* 8(5):284–292
2. Bini E, Buttazzo GC (2004) Schedulability analysis of periodic fixed priority systems. *IEEE Trans Comput* 53(11):1462–1473
3. Bini E, Buttazzo GC, Buttazzo GM (2001) A hyperplanes bound for the rate monotonic algorithm. In: *Proceedings of the 13th euromicro conference on real-time systems*, pp 59–67
4. Bini E, Natale, MD, Buttazzo, G (2008) Sensitivity analysis for fixed-priority real-time systems. *Real-Time Syst* 39(1–3):5–30
5. Burns, A, Wellings, A (2001) *Real-time systems and programming languages*. Ada 95, Real-Time Java and real-time POSIX, 3rd edn. Addison Wesley Longman, Reading
6. Buttazzo GC (2005) Rate monotonic vs. edf: Judgment day. *Real-Time Syst* 29(1):5–26
7. Davis RI, Zabus A, Burns A (2008) Efficient exact schedulability tests for fixed priority real-time systems. *IEEE Trans Comput* 57:1261–1276
8. George L, Riverre N, Spuri M (1996) *Preemptive and non-preemptive real-time uniprocessor scheduling*. Technical report, 2966, INRIA
9. Liu JWS (2000) *Real time systems*, 1st edn. Prentice Hall, New York
10. Joseph M, Pandya P (1986) Finding response times in a real-time system. *Comput J* 29(5):390–395
11. Laplante PA (2004) *Real-time systems design and analysis*, 3rd edn. Wiley, Inc., New York
12. Lehoczky JP, Sha L, Ding Y (1989) The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: *Proceedings of the IEEE real-time system symposium*, pp 166–171
13. Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard real-time environment. *J ACM* 20(1):40–61
14. Manabe Y, Aoyagi S (1998) A feasibility decision algorithm for rate monotonic and deadline monotonic scheduling. *Real-Time Syst* 14(2):171–181
15. Min-Allah N, Wang Y, Jian-Sheng X, Jiu-Xiang L (2007) Revisiting fixed priority techniques. In: *Embedded and ubiquitous computing*. LNCS, vol 4808. Springer, Berlin, pp 134–145
16. Min-Allah N, Ali I, Xing J, Wang Y (2010) Utilization bound for periodic task set with composite deadline. *J Comput Electr Eng* 36(6):1101–1109
17. Min-Allah N, Khan SU, Wang Y (2010) Optimal task execution times for periodic tasks using non-linear constrained optimization. *J Supercomput*. doi:10.1007/s11227-010-0506-z
18. Orozco J, Cayssials R, Santos J, Santos R (1998) On the minimum number of priority levels required for the rate monotonic scheduling of real-time systems. In: *Proceedings of the 10th euromicro workshop on real time system*.
19. Sjodin M, Hansson H (1998) Improved response-time analysis calculations. In: *Proceedings of the 19th IEEE real-time systems symposium*, pp. 399–409