

A HYBRID TEST FOR FASTER FEASIBILITY ANALYSIS OF PERIODIC TASKS

NASRO MIN-ALLAH¹ AND SAMEE ULLAH KHAN²

¹Department of Computer Science
COMSATS Institute of Information Technology
Park Road, Chak Shahzad, Islamabad 44000, Pakistan
nasar@comstats.edu.pk

²Department of Electrical and Computer Engineering
North Dakota State University Fargo
PO Box 6050, Fargo, ND 58108-6050, USA
samee.khan@ndsu.edu

Received May 2010; revised September 2010

ABSTRACT. *In this paper, the issue of impracticality of testing the feasibility of periodic tasks at run time is studied. We provide an exact test, which is a hybrid solution obtained through both inexact (sufficient but not necessary) and exact (necessary and sufficient) schedulability tests. The task set is divided into two subsets such that (a) all tasks in the first group can be scheduled and the feasibility of this part is tested by the inexact test, while (b) the feasibility of the second group that consists of both schedulable and unschedulable tasks is determined by the exact test. The proposed test outperforms existing alternatives from the perspective of execution time, and becomes an ideal candidate for determining feasibility of online systems. In addition, two well-known bounds, namely LL-bound and H-bound, are compared and it is proved that the H-bound is always true when the LL-bound holds.*

Keywords: Real-time systems, Fixed-priority scheduling, Feasibility analysis, Online schedulability test

1. Introduction. When designing hard real-time multitasking systems, one of the most fundamental issues that must be considered is that of timing correctness; i.e., the system must provide predictable behavior under all possible circumstances [1]. A number of scheduling algorithms are presented for scheduling real-time tasks and Rate Monotonic (RM) scheduling is known as the optimal priority assignment technique under fixed priority scheduling class which assigns priorities to tasks based on activation rates; i.e., smaller is the frequency; higher is the priority, while ties are broken arbitrarily.

The Rate Monotonic Analysis (RMA) is a technique that is used to determine schedulability of periodic tasks under the RM scheduling. The RMA can be classified into two broader categories: (a) inexact/imprecise tests (sufficient conditions) and (b) exact/precise tests (necessary and sufficient conditions). For real-time systems demanding low cpu utilization, imprecise tests may be a reasonable choice to determine feasibility. On the other hand, precise tests become inevitable when the workload is high (up to 100%). Because of its polynomial time complexity, imprecise tests are preferred over precise tests. On the contrary, the associated complexity with exact solution is high, which makes the aforementioned techniques impractical for online analysis. Attempts are being made to reduce the average-case time complexity of precise tests [2, 3, 4, 5, 6, 7, 8]. However, the complexity of these tests still remains pseudo-polynomial. Therefore, further research is needed to bridge the existing gap between polynomial and pseudo-polynomial tests.

The idea of partitioning task set in a multiprocessor environment, on the basis of system utilization, is a well studied one and many results can be found in the literature [9, 10, 11, 12, 13, 14]. Inspired from such partitioning, we provide a formulation that addresses the performance issue by splitting the given periodic task set into two subsets, namely (a) the RM-schedulable task set of which the feasibility is confirmed with polynomial tests and (ii) the undecidable task set (consisting of both schedulable and unschedulable tasks) of which the feasibility is determined only through pseudo-polynomial tests. Experimental results for randomly generated task sets show that such arrangements decently reduce the analysis time while the underlying system remains unchanged.

The rest of the paper is organized as follows: we formulate our problem and provide some definitions in Section 2; the main results for obtaining feasibility of subset with inexact test are presented in Section 3, in addition to proving that H-bound is always true when LL-bound holds; in Section 4, we derive a novel algorithm that is obtained by combining available results to determine the feasibility of a task set at run time; experimental results in Section 5 demonstrate the effectiveness of the hybrid solution and finally, we conclude the paper in Section 6.

2. Preliminaries and Problem Formulation. Real-time systems operate on recurring tasks that are released periodically. Every execution of a task τ_i is called a job j_i . In the most generalize model of periodic tasks, a task τ_i is defined by an offset O_i , which defines: (a) the release time of its first job, (b) its worst-case execution time c_i , (c) its period p_i between two successive releases and (d) its relative deadline d_i that defines the time window in which the job must be executed. Assuming that the task phasings are all zero ($\min_{1 \leq i \leq n} O_i = 0$), the release time $r_{i,k}$ of k^{th} job $j_{i,k}$ of a task τ_i is given by $(k-1)p_i$. Let $\Gamma = \{\tau_1, \dots, \tau_n\}$ denote a set of independent periodic tasks. For each task τ_i , the utilization can be given by $u_i = c_i/p_i$. The cumulative utilization of periodic task system τ is defined as:

$$u(n) = \sum_{i=1}^n u_i \quad (1)$$

Our work is targeting systems with independent, preemptable and hard periodic tasks in which at most only one job $j_i \in \tau_i$ exists at any given point in time t . Moreover, all tasks immediately get ready for execution on a uniprocessor as soon as they are released. Furthermore, the deadlines are equal to the periods. In the aforementioned system, we ignore the associated tasks overheads (task swapping times, etc.) and assume that the number of priority levels are unlimited. We also assume that initially all of the tasks arrive simultaneously at critical instant [15]. The following definitions will be used in the following sections of this paper.

Definition 2.1. \mathbf{T} represents a subset of task set τ having priority higher than τ_i .

Definition 2.2. (Arithmetic and Geometric Means Inequality) For a positive a and b , $(a+b)/2 \geq \sqrt{ab}$.

According to our task model, there exist only one job $j_{i,c}$ of any task τ_i at any time t and therefore, for convenience we represent $j_{i,c}$ by j_i and $R_{i,c}$ by R_i . Unlike previous works [3, 16, 17, 18], the periodic task set Γ is bifurcated into τ^{st} and τ^{et} , where $\tau^{st} = \{\tau_1, \tau_2, \dots, \tau_h\}$ for $1 \leq h \leq n$ and $\tau^{et} = \{\tau_{h+1}, \tau_{h+2}, \dots, \tau_n\}$. The former set represents those task whose feasibility can be determined with sufficient condition. The latter set consist of $(n-h)$ lower priority tasks such that any lower priority task $\tau_i \in \tau^{et} = \tau \setminus \tau^{st}$, where $h < i \leq n$. Task τ_i is either schedulable or unschedulable using the RM algorithm. Its feasibility can only be determined with the necessary and sufficient conditions. The

flow chart given in Figure 1, explains the working of our algorithm, where the flow of hybrid formulation is divided into two parts. This combination is represented by the Hybrid Test (HT), in rest of the paper. The inexact part of HT is discussed in Section 3, while Section 4 covers the exact part.

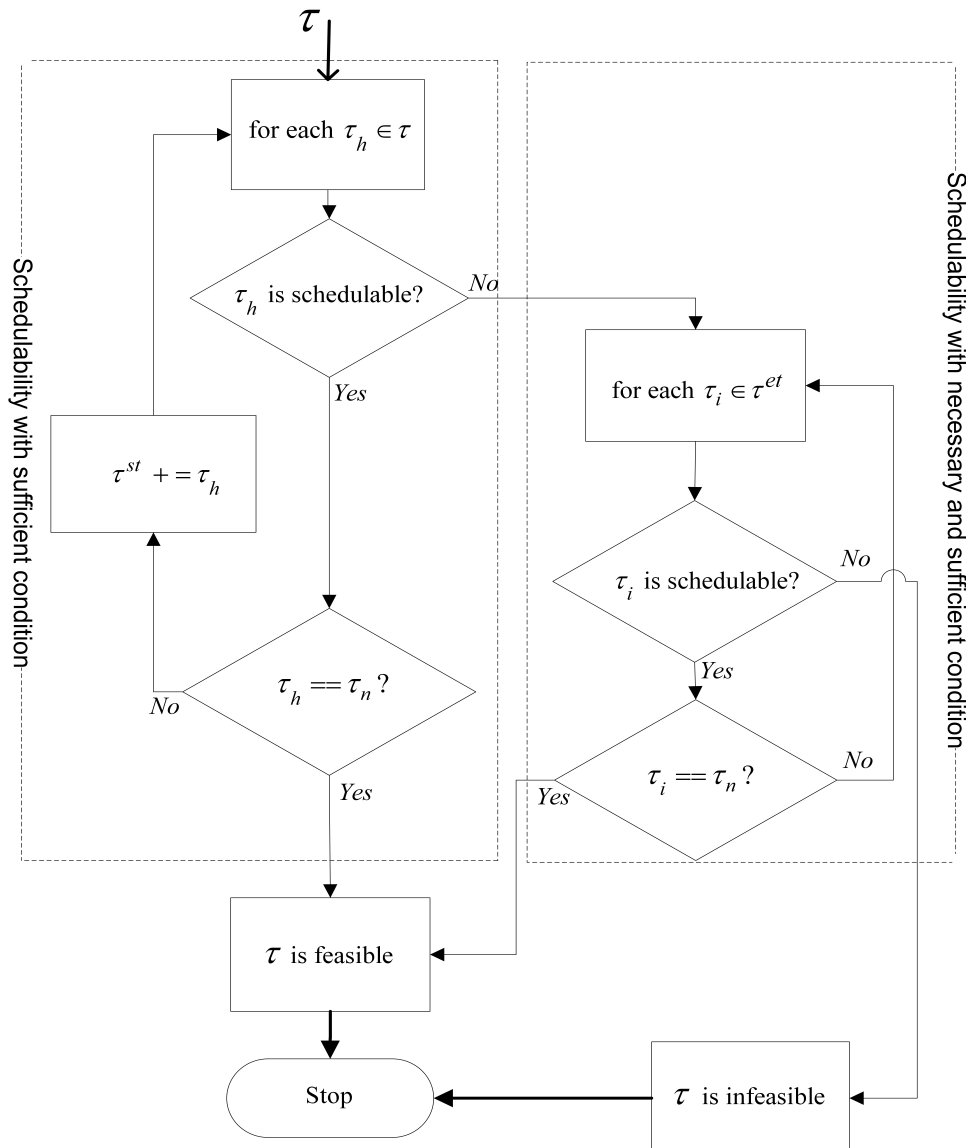


FIGURE 1. Flow chart of the hybrid test

3. **Formation of τ^{st} .** According to the LL-bound [15], a periodic task system is static-priority feasible if

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

where n denotes the number of tasks in Γ . 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 on a preemptive uniprocessor, if the RM algorithm is used and $u(n)$ is not greater than $\ln(2)$. This result gives a simple $O(n)$ procedure to test task feasibility online, where tasks can arrive at run time. However, the aforementioned technique is only a sufficient condition. It is quite possible for an implicit-deadline synchronous periodic task system to exceed the LL-bound and still be classified RM-feasible.

A better utilization based test, termed Hyperbolic bound (H-bound) is proposed in [19].

Theorem 3.1. *Let $\Gamma = \{\tau_1, \dots, \tau_n\}$ be a set of n periodic tasks, where each task τ_i is characterized by a processor utilization u_i . Then, Γ is schedulable with the RM algorithm if [19]:*

$$\prod_{i=1}^n (u_i + 1) \leq 2 \quad (3)$$

In the following, we show that the H-bound is always true when the LL-bound holds. However, we provide a counterexample in Table 1 to show that the converse does not hold.

Theorem 3.2. *The H-bound $\prod_{i=1}^n (u_i + 1) \leq 2$ is always true if*

$$\sum_{i=1}^n u_i \leq n(2^{1/n} - 1).$$

Proof:

$$\begin{aligned} \text{Because } \sum_{i=1}^n u_i &\leq n(2^{1/n} - 1) = u_1 + u_2 + \dots + u_n \leq n(2^{1/n} - 1) \\ &= (1 + u_1) + (1 + u_2) + \dots + (1 + u_n) \leq n(2^{1/n}) \\ &= \left\{ \frac{(1 + u_1) + (1 + u_2) + \dots + (1 + u_n)}{n} \right\}^n \leq (2^{1/n})^n \end{aligned}$$

By Definition 2.2:

$$\begin{aligned} &= (u_1 + 1) \times (u_2 + 1) \dots (u_n + 1) \leq \left\{ \frac{(u_1 + 1) + (u_2 + 1) + \dots + (u_n + 1)}{n} \right\}^n \\ &= (1 + u_1) \times (1 + u_2) \dots (1 + u_n) \leq (2^{1/n})^n = 2 \end{aligned}$$

As $\prod_{i=1}^n (u_i + 1) = (1 + u_1) \times (1 + u_2) \dots (1 + u_n)$,

which completes the proof. \square

In the following, we show by a counterexample that the converse for Theorem 3.2 is not true. Consider the task set given in Table 1: (a) that has $u(n) = 0.750$, (b) whose schedulability is analyzed in descending priority order and (c) task schedulability by both LL-bound and H-bound are recorded in the respective columns.

TABLE 1. Task-level and set-level RM-feasibility with LL-bound and H-bound

τ_i	c_i	p_i	u_i	LL-bound	H-bound
1	30	100	0.300	✓	✓
2	15	125	0.120	✓	✓
3	30	140	0.214	✓	✓
4	7	170	0.041	✓	✓
5	15	200	0.075	inconclusive	✓

Because:

- $(u(n) = 0.7505) > (n(2^{1/n} - 1) = 0.743)$. Although LL-bound can positively determine the RM schedulability of the first four tasks, it fails to answer the RM feasibility of the entire task set.

- $(\prod_{i=1}^n (u_i + 1) = 1.9789) < (2)$. According to H-bound, the task set is RM feasible.

Though the advantage of H-bound over LL-bound is clear, it is worth to note that for an arbitrary n , the inequality (3) becomes $(1 + u(n)/n)^n \leq 2$, when the utilization of all tasks are equal. For such a utilization, the H-bound becomes the same as LL-bound $u(n) \leq n(2^{1/n} - 1)$ [22].

For higher utilization, task parameter tuning is a popular technique. [22] proposed an utilization based test for a restricted task set ($d_i \geq p_i$) that is both necessary and sufficient in nature.

Theorem 3.3. *Consider a system Γ of periodic, independent and preemptable tasks. Assume that the tasks have relative deadlines that are at least equal to their periods. The task set is schedulable on a uniprocessor by the RM algorithm if and only if, the total utilization is at most one.*

Following the naming convention, we denote this bound by L-bound (Named after the conceiver J. W. S. Liu [22]). In contrast to other utilization tests, the L-bound is the only exact test in this category. Although the L-bound offers the maximum bound, the bound is only applicable to a special type of task set. We compare the L-bound, H-bound and LL-bound in Table 2.

TABLE 2. Supremacy of L-bound for restricted task set

τ_i	c_i	p_i	u_i	LL-bound	H-bound	L-bound
1	2	3	0.666	✓	✓	✓
2	1.5	6	0.250	inconclusive	inconclusive	✓
3	0.5	12	0.041	inconclusive	inconclusive	✓
4	1	24	0.041	inconclusive	inconclusive	✓

As indicated by the L-bound, the task set is RM feasible. This notion is illustrated in Figure 2, where RM schedule of tasks is drawn for the first hyperperiod that starts at the critical instant [15] $t = 0$ up to $t = 24$ (the deadline of τ_4). For periodic tasks $\lceil p_i/p_j \rceil = \lfloor p_i/p_j \rfloor$ for $j = 1, 2, \dots, i$. Therefore, all $\lceil p_i/p_j \rceil$ job belongs to any task τ_j released in interval $[0, 24]$ are scheduled according to RM priority order. This implies that the task set is RM feasible. In Table 2, the system utilization is set to the maximum ($u(n) = 1$) and we observe that except L-bound, both LL-bound and H-bound fail. This is because:

- For LL-bound,

$$(u(n) = 1) > (n(2^{1/n} - 1) = 0.75).$$

- For H-bound,

$$(\prod_{i=1}^n (u_i + 1) = 2.56) > 2.$$

Similarly, the seminal work of Liu and Layland is extended in [21, 22] by making use of some additional task parameters. All of aforementioned tests are Sufficient Conditions (SC) and provide faster analysis at the expense of heavy processor utilization.

3.1. Feasibility of τ^{st} with inexact tests. Because we intend to extract the maximum number of tasks from Γ with sufficient condition so that none or few tasks are left to be determined with exact test. Considering the robustness and complexity of utilization based tests [3, 20, 21, 22, 23], we opt for the H-bound. We base our choice on the fact that the H-bound has the same complexity class as that of the LL-bound but better acceptance ratio, i.e., up to $\sqrt{2}$ for a (large) task set [19, 21].

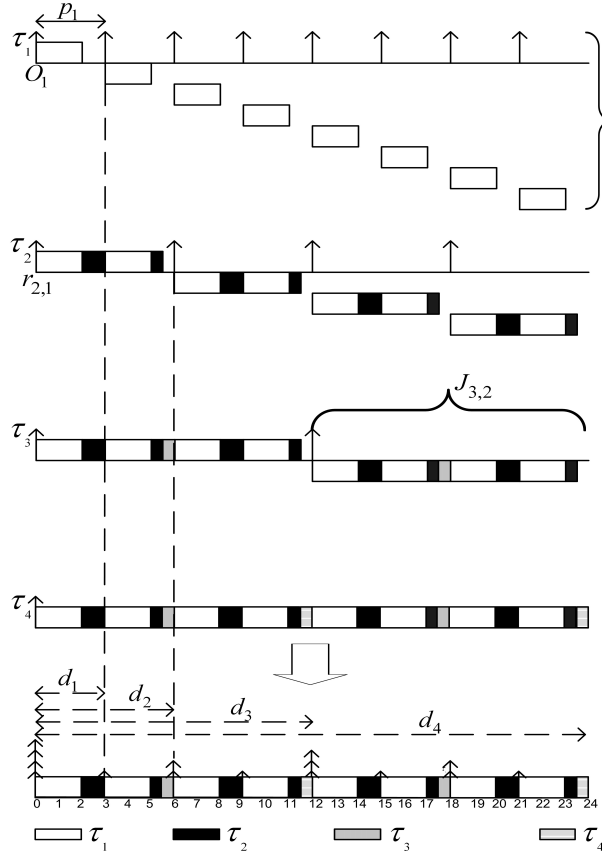


FIGURE 2. RM scheduling of simply periodic task set [22] given in Table 2

4. Feasibility Analysis of τ^{et} with Exact Tests. Existing RMA techniques that encompass both the necessary and sufficient conditions are mostly of pseudo-polynomial time complexity. These tests can be classified in two classes, which are detailed in Sections 4.1 and 4.2, respectively.

4.1. Scheduling point tests. To find schedulability of a task τ_i , the concept of workload was introduced by authors in [20]. The workload constituted by τ_i at time t consists of its execution demand c_i as well as the interference it encounters due to higher priority tasks from τ_{i-1} to τ_1 , which can be expressed mathematically as:

$$w_i(t) = c_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil c_j. \quad (4)$$

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

$$L_i = \min_{0 < t \leq p_i} (w_i(t) \leq t). \quad (5)$$

To verify that such a t exist, Equation (4) is tested at all points in S_i as:

$$S_i = \{ap_b | b = 1, \dots, i; a = 1, \dots, \lfloor p_i/p_b \rfloor\}. \quad (6)$$

From literature, we have the following fundamental theorems to determine whether an individual task and a task set is feasible.

Theorem 4.1. [20] *Given a set of n periodic tasks τ_1, \dots, τ_n , a task τ_i can be feasibly scheduled for all task phasings using the RM algorithm if and only if*

$$L_i = \min_{t \in S_i} \frac{w_i(t)}{t} \leq 1. \quad (7)$$

Theorem 4.2. [20] *The entire task set Γ is feasible if and only if*

$$L = \max_{1 \leq i \leq n} L_i \leq 1 \quad (8)$$

From Equation (7), we gather that L_i is needed to be analyzed only at a finite number of points. The first scheduling point $t \in S_i$ that satisfies the conditions of Equation (4) for any task τ_i determines the schedulability of τ_i at time t . For any task τ_i , the number of elements in set S_i is of particular interest. Every element of S_i contributes a RM scheduling point and constitutes an inequality for τ_i . The number of elements in S_i becomes large especially when p_n/p_1 is large [3]. Clearly, a more efficient technique would aim to restrict the number of scheduling points.

To reduce scheduling points, [3] provides a formulation, termed Hyper-planes Exact Test (HET). The HET technique reduces the scheduling point for τ_i from the original set S_i to a reduced set $H_i(t)$. For any task τ_i , the test begins with p_i and expands its search space by

$$H_i(t) = H_{i-1} \left(\left\lfloor \frac{t}{p_i} \right\rfloor p_i \right) \cup H_{i-1}(t), \quad (9)$$

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

4.2. Fixed-point techniques. As mentioned previously, one method for avoiding all t 's within the range $[0, p_i]$ is to test τ_i 's schedulability at $t \in S_i$ (for RMA) or $t \in H_i$ (for HET). The other alternative is by iteration. To do so, the worst-case workload of τ_i within $[0, p_i]$ can be expressed as

$$t = c_i + \sum_{j \in \mathbf{T}} \left\lceil \frac{t}{p_j} \right\rceil c_j \quad (10)$$

To compute R_i that is equal to the smallest value of t , which satisfies the above equation. [4, 5] proposed an iterative method to solve Equation (10), termed Response Time Analysis (RTA). To find R_i , Equation (10) can be solved with the fixed-point iteration, starting from an initial guess R_i^0 . Because the maximum response time R_i of a task τ_i is at least equal to its execution time, R_i^0 must be equal to c_i . Let $R_i^{\#n}$ be the n -th approximation to the true value of R_i . During the l -th iteration for $l \geq 1$, $R_i^{\#l+1}$ can be computed as:

$$R_i^{\#l+1} = c_i + \sum_{j \in \mathbf{T}} \left\lceil \frac{R_i^{\#l}}{p_j} \right\rceil c_j, \quad (11)$$

which converges after a finite number of iterations. The aforementioned phenomena is due to the fact that 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 value of l or when $R_i^{\#l+1} > d_i$ (whichever occurs first). In the former case, τ_i is schedulable ($R_i \leq d_i$), while in the latter case τ_i is not schedulable. An improvement to RTA is proposed in [5], termed Response Time Improved (RTI) that utilizes R_{i-1}^0 for assigning initial values to R_i . That is, $R_i^0 = R_{i-1}^0 + c_i$ and for this reason alone RTA converges faster compared to RTA.

4.3. Adopting HET for exact part. As the main objective of this paper is to simplify the feasibility analysis process, the feasibility of τ^{et} must be done in an efficient manner. [3] mentions that the HET procedure is the most efficient compared to other exact tests. Therefore, we also choose the HET procedure to determine the feasibility of τ^{et} .

4.4. The worst case. In this section, we discuss the worst case behavior of our proposed testing procedure. In the worst case, there may exist task sets whose high i -priority tasks $(\tau_1, \tau_2, \dots, \tau_i)$ present the maximum utilization such that $u(i) > i(2^{1/i} - 1)$ or $\prod_{j=1}^i (u_j + 1) > 2$. In this case, the inexact part of HT is unable to accommodate a majority of the tasks. Because all of the $n - i$ tasks must be determined with the exact part, the test becomes an exact test. Although one may argue that such a scenario is very unlikely for a random task set, we consider this to be the worst case scenario. To tackle the worst case scenario, we propose the following two approaches.

1. For the inexact part, sort tasks in the ascending order keyed by their utilization factor such that $\tau^{st} \neq \emptyset$. For the exact part, the same priority order is kept so that the RM-priority is maintained. We must note that sorting incurs an additional overhead.
2. Based on the designer's knowledge, the utilization factor of any task can be restricted to reflect the load of an application. Even for a randomly generated task set $\tau^{st} > \tau^{et}$ the eventual analysis time is reduced with the proposed hybrid approach.

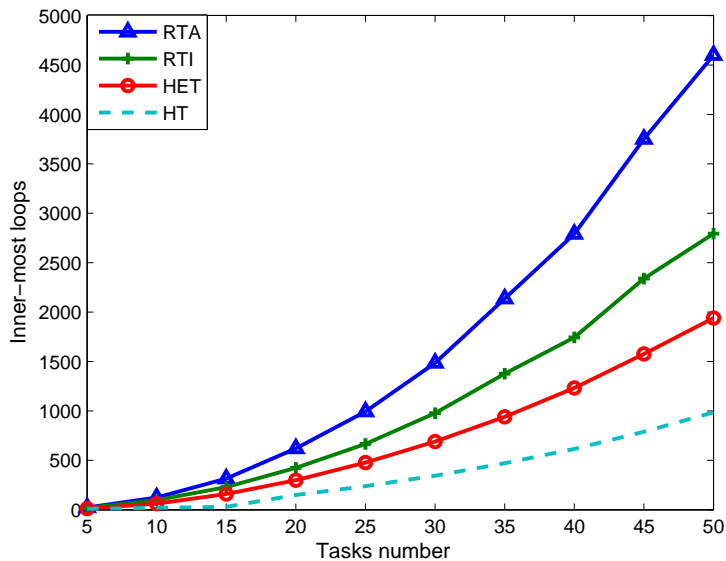


FIGURE 3. Supremacy of HT for larger task sets

5. Experimental Analysis. In this section, we evaluate the performance of the proposed HT methodology. To make a fair comparison, the other three exact tests namely, RTA, RTI, and HET are implemented and observed. For our experiments, we generate task periods on random from within the range of $[10, 100000]$ of which the elements are uniformly distribution. Similarly, for task execution demands, random values are generated from a uniform distribution with a range of $[1, p_i]$. We deliberately use uniform distribution so that the worst case is encountered by the H-bound. Tasks priorities are assigned according to the RM algorithm. Results are obtained after an average of 500 iterations of a task set of variable sizes (5 to 50) that are incremented by a step of 5.

[19] suggests that one of the most suitable criteria for comparisons among these tests is to record the number of the innermost loops executed by an exact test before it concludes feasibility (see [3] for details).

When the system utilization is low $u(n) \leq \ln(2)$, the feasibility of HT is decided with the sufficient part, i.e., $\tau^{st} = \tau$ and $\tau^{et} = \emptyset$. In such a case, the number of the innermost loop executed by HT is a function of the task set size.

When the system utilization is high, there is a high possibility that some of the tasks may not be determined with exact part of HT. Therefore, more of the innermost loops must be executed. This higher utilization shows that the maximum number of tasks are deemed feasible with the exact part of HT, i.e., $\tau^{et} > \tau^{st}$. If the utilization is further increased, then the task set is deemed infeasible. The first infeasible task τ_i must be determined early so that the rest of the $n-i$ tasks can be skipped to reduce the complexity, which is the forte of the HT technique. This is evident from Figure 3, that illustrates the effectiveness of the HT procedure by executing lesser number of inner loops as the number of tasks increase within the system.

6. Conclusion. This paper proposed a hybrid feasibility test, termed HT. The HT technique integrated the Sufficient Conditions (SC) and the Necessary and Sufficient Conditions (NSC) for faster feasibility analysis. The HT technique is most suitable for analyzing the online feasibility of hard real-time periodic systems under fixed priority scheduling.

Experimental result showed that the HT approach was much faster compared to existing techniques even when the system had 100% utilization. As a future work, an interesting extension of the current work would be to address the non-preemptive counterpart of the studied problem with the HT approach.

REFERENCES

- [1] H. C. R. Ha and J. W. S. Liu, Experimental analysis of timing validation methods for distributed real-time systems, *Journal of Supercomputing*, vol.25, no.1, pp.2169-2174, 2003.
- [2] L. George, N. Riverre and M. Spuri, Preemptive and non-preemptive real-time uniprocessor scheduling, *Research Report 2966*, INRIA, France, 1996.
- [3] E. Bini and G. C. Buttazzo, Schedulability analysis of periodic fixed priority systems, *IEEE Transactions on Computers*, vol.53, no.11, pp.1462-1473, 2004.
- [4] N. C. Audsley, A. Burns, K. Tindell and A. Wellings, Applying new scheduling theory to static priority preemptive scheduling, *Software Engineering Journal*, vol.8, no.2, pp.80-89, 1993.
- [5] M. Sjodin and H. Hansson, Improved response-time analysis calculations, *Proc. of the 19th IEEE Real-Time Systems Symposium*, pp.399-409, 1998.
- [6] S. Baruah and K. Pruhs, Open problems in real-time scheduling, *Journal of Scheduling*, vol.13, no.6, pp.577-582, 2010.
- [7] N. Min-Allah, S. U. Khan, N. Ghani, J. Li, L. Wang and P. Bouvry, A comparative study of rate monotonic schedulability tests, *Journals of Supercomputing*, 2011.
- [8] T. W. Kuo, L. P. Chang, Y. H. Liu and K. J. Lin, Efficient online schedulability tests for real-time systems, *IEEE Transactions on Software Engineering*, vol.29, no.8, pp.734-751, 2003.
- [9] J. M. Lopes, J. L. Diaz and D. F. Garcia, Minimum and maximum utilization bounds for multiprocessor rate monotonic scheduling, *IEEE Transaction on Computers*, vol.15, no.7, pp.642-653, 2004.
- [10] Y. Oh and S. H. Son, Allocating fixed-priority periodic tasks on multiprocessor systems, *Real-Time Systems*, vol.9, no.3, pp.207-239, 1995.
- [11] D. Oh and T. P. Baker, Utilization bounds for N-processor rate monotonic scheduling with static processor assignment, *Real-Time Systems*, vol.15, no.2, pp.183-193, 1998.
- [12] N. Fisher, S. Baruah and T. P. Baker, The partitioned scheduling of sporadic tasks according to static-priorities, *Proc. of the 18th Euromicro Conference on Real-Time Systems*, pp.118-127, 2006.
- [13] T. Ma, Q. Yan, D. Guan and S. Lee, Research on task scheduling algorithm in grid environment, *ICIC Express Letters*, vol.4, no.1, pp.1-6, 2010.

- [14] W. Jia, B. Han, C. Zhang and W. Zhou, Delay control and parallel admission algorithms for real-time anycast flow, *Journal of Supercomputing*, vol.29, no.2, 2004.
- [15] C. L. Liu and J. W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, *Journal of the ACM*, vol.20, no.1, pp.40-61, 1973.
- [16] N. Min-Allah, I. Ali, J. Xing and Y. Wang, Utilization bound for periodic task set with composite deadline, *Journal of Computers and Electrical Engineering*, vol.36, no.6, pp.1101-1109, 2010.
- [17] N. Min-Allah, S. U. Khan and Y. Wang, Optimal task execution times for periodic tasks using nonlinear constrained optimization, *Journal of Supercomputing*, 2010.
- [18] R. I. Davis, A. Zabos and A. Burns, Efficient exact schedulability tests for fixed priority real-time systems, *IEEE Transactions on Computers*, vol.57, pp.1261-1276, 2008.
- [19] E. Bini, G. C. Buttazzo and G. Buttazzo, A hyperbolic bound for the rate monotonic algorithm, *Proc. of the 13th Euromicro Conference on Real-Time Systems*, pp.59-66, 2001.
- [20] J. P. Lehoczky, L. Sha and Y. Ding, The rate monotonic scheduling algorithm: Exact characterization and average case behavior, *Proc. of the IEEE Real-Time System Symposium*, pp.166-171, 1989.
- [21] E. Bini, G. C. Buttazzo and G. Buttazzo, Rate monotonic analysis: The hyperbolic bound, *IEEE Transactions on Computers*, vol.52, no.7, pp.933-942, 2003.
- [22] J. W. S. Liu, *Real Time Systems*, Prentice Hall, 2000.
- [23] I. Imura, Y. Moriyama and S. Nakayama, Consideration on distributed immune algorithm in job-shop scheduling problem, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.5003-5010, 2009.