ICIC Express Letters ICIC International ©2018 ISSN 1881-803X
Volume **12**, Number **8**, August **2018** pp. 759–766

AN IMPROVED HYBRID TEST FOR FEASIBILITY ANALYSIS OF PERIODIC TASKS

Saleh Alrashed

Deanship of Admissions and Registration
Imam Abdulrahman Bin Faisal University
P.O. Box 1982, Dammam, Saudi Arabia
saalrashed@uod.edu.sa

Received January 2018; accepted April 2018

Abstract. *Realizing the NP-completeness of the exact conditions for feasibility analysis of hard real-time systems, there has been a growing interest in reducing the computational cost of such techniques. In this paper, we propose a hybrid solution that is derived by combining the existing necessary and sufficient-only conditions. The method splits the task set into two parts and then determines feasibility of each part accordingly. Similar techniques have been proposed recently but unlike existing approaches, the second part of feasibility analysis is tested through necessary and sufficient condition with lowest priority first approach. Our experimentation results are aligned with theoretical formulation and show that our hybrid solution presents significant improvement over existing algorithms from performance perspective, especially when the system demands a higher CPU utilization.*

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

1. **Introduction.** Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computations, but also on the time at which the results are produced [1, 2]. Due to implicit nature of period tasks with the timing constraints and target domain, especially under hard real-time systems, the schedulability analysis of such systems demands mathematical proven solutions and cannot be left to statistical or average case analysis.

For predictable systems, fixed priority of a task is always preferred. Using fixed priority assignment, the highest priority task never misses its deadline when the system is overloaded [1, 2, 4, 5]. For the systems where predictability is more important than performance, fixed priority scheme is being used. Typical examples missile guidance system, flight control systems, avionic control, power plants, automotive, medical devices, robotics and organizations such as US-DoD, IBM, and General Motors [6, 7]. On the contrast, dynamic algorithms are considered better theoretically [8, 9]; however, this priority assignment to individual tasks results in unpredictable behavior in overloaded scenarios [2, 14, 15, 16, 17]. In this paper, we present a fixed-priority scheduling algorithm that assigns the same priority to all jobs belonging to a task under uni-processor systems.

Rate Monotonic (RM) is the most predictable scheduling algorithm under fixed priority scheduling. To answer the schedulability of a task set under RM scheduling algorithm, the task set is subject analysis of feasibility tests. The task set is always RM schedulable if it passes the test; otherwise, the set is declared infeasible. These feasibility tests are divided into two main types: (i) Sufficient Condition (SC), (ii) Necessary and Sufficient Condition (NSC). Till date, the complexity of SC is polynomial while NSC is pseudo polynomial. Unfortunately, SC can only answer the feasibility of the system having 69%

DOI: 10.24507/icicel.12.08.759

utilization while NSC can answer feasibly of a system as long as utilization is not more than 100%; however, the associated complexity is very high. Recently, the focus has been made on lowering the computation cost of RM feasibility test by combining necessary and sufficient condition for faster feasibility tests.

Since RM algorithm maps the highest priority level to the task with shortest period and gives the highest privity to the most critical task in the system. Under preemptive scheduling, RM is fixed priority scheduling algorithm and CPU is always allocated to the higher priority tasks when a decision has to be made. Nevertheless, the priority of tasks always reflects which task will miss the deadline whenever the system becomes infeasible. Authors in [20] established a mechanism for the faster feasibility analysis based on the fact that if the system is infeasible, it is mainly because of the lower priority tasks and not the highest priority, ablest. The work in [20] answers the feasibility of the system much faster, especially when the system presents higher CPU demand.

Traditionally, the feasibility of the real-time system is determinedly in the priority order, i.e., the feasibility of highest priority task is tested first and then the next task with one level lower priority is tested and so on. However, the lower priority task potentially makes the system infeasible and hence, authors in [10] established a formulation that integrated the existing solution for faster feasibility analysis. We extend the work done in [10] by determining the feasibility with NSC by determining the feasibility of the subset of tasks using lowest priority first approach.

We propose a novel test for a faster feasibility test that guarantees 100% CPU utilization by splitting the task set into two subsets. The first subset's feasibility is determined with "Inexact Condition (LL-bound [5])", while the reaming task set's feasibility is subject to "Exact Condition". As a main contribution of this work, we test schedulability of the task set in ascending priority order starting from the lowest priority task in the system for the 2nd subset. This arrangement results in a faster feasibility analysis of the systems as the 2nd subset is undertaken by exact condition which is pseudo-polynomial. The proposed technique is evaluated under various system utilization ranging from 70 to 100% for the task set of size 5 to 50 and the results obtained are encouraging.

We divide rest of the paper into 4 sections. Section 2 introduces the preliminaries and presents the task model for the problem formulation. The main results are presented in Section 3, while experimental results are discussed in Section 4. Finally, the paper is concluded in Section 5.

2. Preliminaries and Problem Formulation.

Before we present the main contribution of this paper, we provide notations in Table 1 and definition below.

Before proceeding on a more detailed description, we first provide a few definitions and notations that are used frequently in rest of the paper.

Definition 2.1. (*Critical Instant* [5]) *In fixed-priority systems, every job $j_{i,c}$ must complete before the next job $J_{i,c+1}$ of the same task τ_i. A critical of any task τ_i occurs when one of its jobs $j_{i,c}$ is released at the same time as a job of a higher-priority task, i.e., $r_{i,c} = r_{k,l_k}$ for some l_k for every $k = 1, 2, \ldots, i-1$.*

Definition 2.2. (*Higher Priority Subset*) **A** *represents a subset of task set τ having a priority higher than τ_i.*

Definition 2.3. (*Lower Priority Subset*) **B** *represents a subset of task set such that* $\mathbf{B} = \tau - \mathbf{A}$ *and not answerable by LL-bound.*

Definition 2.4. (*Simply Periodic Task Set* [1]) *A system of periodic tasks is simply periodic (harmonic) if the period of each task is an integer multiple of the period of the other tasks. That is, $p_k = np_i$, where $p_i \le p_k$ and n is a positive integer ($\forall\tau_i$ and $\forall\tau_k$).*

Table 1. Notations

Notation	**Meaning**
τ	The set of periodic tasks
$J_{i,k}$	k-th job of i-th task, $\tau_i \in \tau$
E_i	Worst case execution time of τ_i
P_i	Period of τ_i
D_i	Deadline of τ_i
U_i	Utilization of τ_i
n	Number of elements in Γ
$U(N)$	Utilization of Γ
$W_i(t)$	The cumulative workload due to τ_i at time t
$priority(\tau_i)$	Priority of τ_i
$\mathbf{T_i}$	Subset of task set having a priority higher than τ_i
R_i	Maximum response time of τ_i
S_i	Set of scheduling points for τ_i
$H_i(t)$	Set of scheduling points for τ_i where $H_i(t) \subseteq S_i$
L_i	Schedulablility condition for τ_i
L	Feasibility condition for τ

Definition 2.5. (*Maximum Response Time* [26]) *The longest time ever taken by an instance of a task from its release time until the time it completes its required computation. For $j_{i,c}$ this time is denoted by $R_{i,c}$.*

In our problem formulation, we assume a period task set having N tasks where each task has a unique priority. Each task τ_i is represented by (E_i, P_i), where E_i is executions time and P_i is the task period. All tasks arrive at critical instant and the maximum response time of each job is less than or equal to the deadline. RM [5] policy is used for priority assignment such that for any two tasks τ_i and τ_j, priority $(\tau_i) >$ priority $(\tau_j) \Rightarrow$ period $(\tau_i) <$ period (τ_j), while ties are broken arbitrarily. RM is the optimal static priority scheduling algorithm for implicit-deadline model (when deadlines coincide with respective periods) [7, 11, 18, 19, 21, 22, 23]. The utilization of task τ_i is shown by $U_i = E_i/P_i$. The cumulative utilization of periodic task system τ is defined as:

$$U(N) = \sum_{i=1}^{N} U_i \tag{1}$$

The first feasibility test for scheduling the above formulated system under fixed-priority scheme on single processor systems was first addressed by Liu and Layland [5] in 1973. They derived the optimal static priority scheduling algorithm for implicit-deadline model (when deadlines coincide with respective periods) called Rate Monotonic (RM) algorithm. The test in [5] is called LL-bound [5], and according to LL-bound a periodic task system of independents is RM schedulable if

$$U(N) \leq N\left(2^{1/N} - 1\right) \tag{2}$$

When $N \to \infty$, $N\left(2^{1/N} - 1\right)$ becomes $\ln(2)$. So any periodic system with utilization less than 69% is RM feasible on single processor system. It can be seen from Equation (2) that there are just N-iterations needed for determining the feasibility of the system and hence the complexity is polynomial. An associated disadvantage is losing $1 - \ln(2)\%$ utilization. More techniques such as simply periodic task set can result in up to 100% utilization but the constraints in that the task periods need to be in harmonic fashion. To fix the utilization problems NSC conditions were prospered in [10, 14, 15, 16, 26, 27, 28].

3. **Improved Feasibility Test.** The workload due to τ_i at time t has two parts: (i) execution demand E_i of the task $w_i(t)$ and (ii) execution demands at the same point t due to all higher priority tasks. In other words, the workload $w_i(t)$ of τ_i is expressed as:

$$w_i(t) = E_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{P_j} \right\rceil E_j \tag{3}$$

Task τ_i is only schedulable at any time t when

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

Equation (3) is tested at all points S_i as

$$S_i = \{ap_b | b = 1, \ldots, i; a = 1, \ldots, \lfloor P_i/P_b \rfloor\} \tag{5}$$

The first exact condition known as Time Demand Analysis (TDA) was introduced in [26] called TDA for RM feasibility analysis.

Theorem 3.1. [26] *Given a set of n periodic tasks $\tau_1, \ldots, \tau_n$, a task τ_i can be feasibly scheduled for all task phasing using the RM algorithm if and only if*

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

Theorem 3.2. [26] *The task set is RM schedulable iff:*

$$L = \max_{1\leq i\leq N} L_i \leq 1 \tag{7}$$

To reduce the number of scheduling points in TDA, [12] derived Hyper-planes Exact Test (HET). The HET suggests that a subset of points is enough to check the schedulability of any task. This solution works on lowering original set S_i to a reduced set $H_i(t)$. According to HET, the schedulability test of task $\tau_i t$ 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 $H_0(t) = \{t\}$.

Similarly, Enhanced Rate Monotonic Time Demand Analysis (ETDA) was introduced recently to further enhance the TDA. The ETDA helps avoid testing the feasibility of tasks at unnecessary points during feasibility analysis by:

$$L_i = \min_{t\in S_Z} \frac{w_i(t)}{t} \leq 1 \tag{9}$$

where $Z_i = S\_X_{i-1}$ and X_{i-1} is the set of scheduling points at which the schedulability of τ_{i-1} is false.

Like [10], the feasibility of **A** to be tested with LL-bound and the remaining lower priority task **B** are subject to the application of exact condition. We represent the work done in [10] by Hybrid Test (HT). For the **B**, we use the lowest priority first approach [20]. **B** has many tasks and the unitization of these tasks is $\sum_{\mathbf{B}} U_i$. However, LL-bound is not applicable for this part as **A** is already addressed with LL-bound. Eventually, **B** has to be tested with NSC. Again, the issue is whether to use this part with highest priority first or lowest priority first technique. All the tasks in the **B** have lower priorities and it is very likely that the cumulative demand of τ is higher and that is why only **A** was addressed with LL-bound. In this situation, we opt for the lowest priority first approach as it concludes the feasibility of **B** faster than the highest priority first counterpart. We denote the utilization of **A** by $U(A)$. In the remaining part of this paper, we represent our work by Improved Hybrid Analysis (IHA) and explain its working in Algorithm 1. It

Algorithm 1.

```
procedure IHA($\tau$)
if ($U(N) \leq 0.69$) then
  $\tau$ is schedulable;
  exist();
else
  $\mathbf{B} = \tau - \mathbf{A}$: ($U(A) \leq 0.69$) // Tasks are schedulable when utilization is below 69%
  as // per LL-bound
end if
for all $\tau_i \in \mathbf{B}$: shorter periods have higher order do
  if ($L_i \leq 1$) then
    $\tau_i$ is schedulable;
  end if
end for
if ($L \leq 1$) then
  $\mathbf{B}$ is feasible;
else
  $\mathbf{B}$ is infeasible;
end if
end procedure
```

can be seen in Algorithm 1 that the feasibility of the task set is influenced by size of **B** and overall system utilization.

4. **Experimental Evaluation.**

In this section, we compare our work with two relevant feasibility methods namely TDA and HT. We generate random task set of size $5, 10, 15, \ldots, 50$ with a step of 5 tasks. For each task τ_i, execution time E_i and task periods P_i of an individual task τ_i are obtained with uniform distribution pertaining the task periods of tasks. We implicitly assume here that task deadlines are in agreement with task periods and task priorities are assigned as per RM policy.

In Figures 1-4, we compare the run time of IHA with the existing exact feasibility analysis techniques. The experimentation is done on Intel Xeon machine with 12MB L3-cache and 48GB memory. For each run, we keep the system utilization at 70%, 80%, 90%, and 100% for evaluating these tests. We measure the time in millisecond in the figures given below. To have confidence in our results, we run each experiment 500 times for the same task set. The x-axis shows the size of the tasks in the system while y-axis represents the time taken by each test.

It can be seen that the time taken by all feasibility tests (TDA, HT and IHA) is quite low, especially for HT and IHA. The reason behind this behavior is TDA only relying on NSC and hence all tasks need to pass through exact feasibility analysis, while HT and IHA also integrated the SC and hence converge much due to low complexity of SC. This is due to the reason that the utilization is below the permissible LL-bound and hence it is applicable in this scenario. With increased task set size, the graphs are still low and the number of tasks becomes irrelevant here. When the utilization is increased, then more tasks need to be anted by the NSC. It can be seen from Figures 1-4 that more utilization increases all tests need more time to determine feasibility of the task. TDA analyzes all tasks and now the task periods of the tasks are larger and hence more scheduling points have to be tested before answering the feasibility of the task. However, the IHA also suppresses HT as IHA is investigating the set with NSC but using the lowest priority first approach and hence the performance is better than existing counterparts. From Figures 3 and 4, it can be seen that IHA concludes the feasibility much faster. The explanation

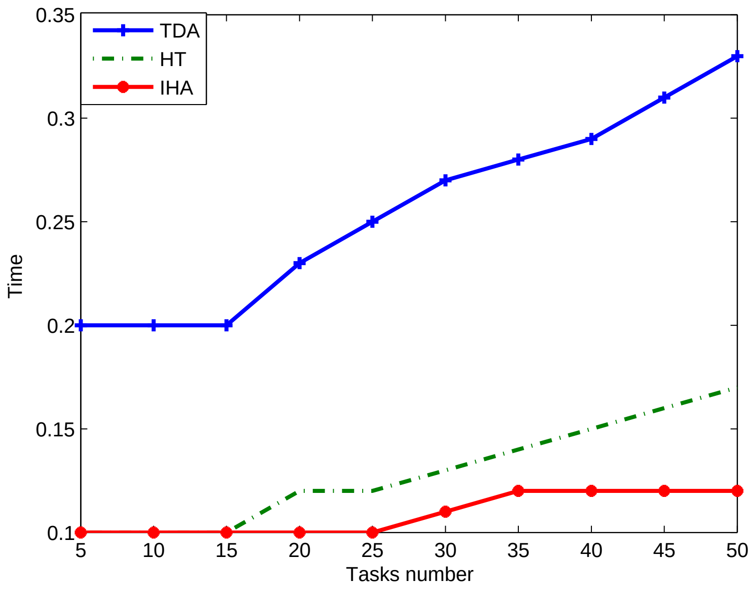

Figure 1. Performance at 70 percent system utilization

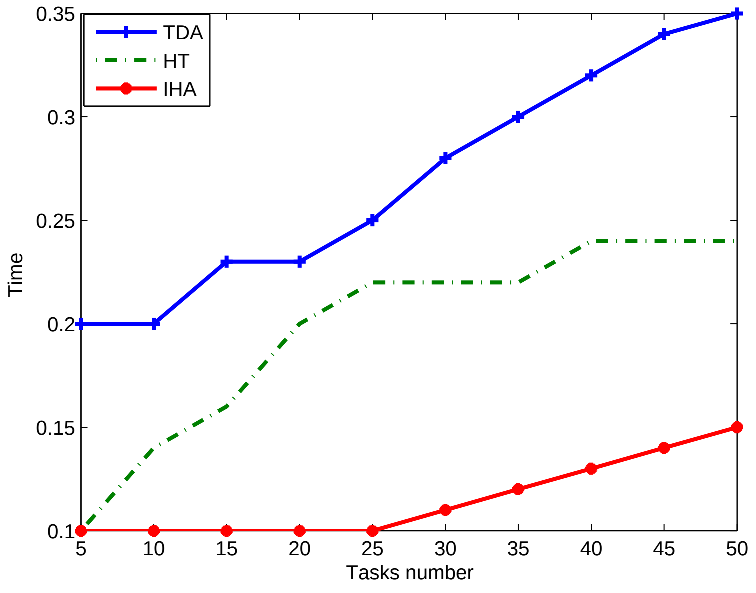

Figure 2. Performance at 80 percent system utilization

for this behavior is that real-time system under RM scheduling is hard to be scheduled on single processor system with higher utilization such as 90% or 100% and hence the system becomes infeasible. As discussed in Section 3, lower priority tasks are more likely to miss the deadline in such cases and hence the feasibility analysis with lowest priority first is concluded much faster. The results obtained in Figures 3 and 4 are in accordance with our observation that **B** becomes infeasible with higher system utilization and hence IHA exhibits better performance with higher system utilization.

5. **Conclusion.** We explored a new dimension of feasibility analysis of hard real-time systems by splitting a periodic task set into two subset. This arrangement helped lower the computation cost of feasibility analysis under RM scheduling algorithm without compromising the timing constraints of the system. The RM feasibility of first part was

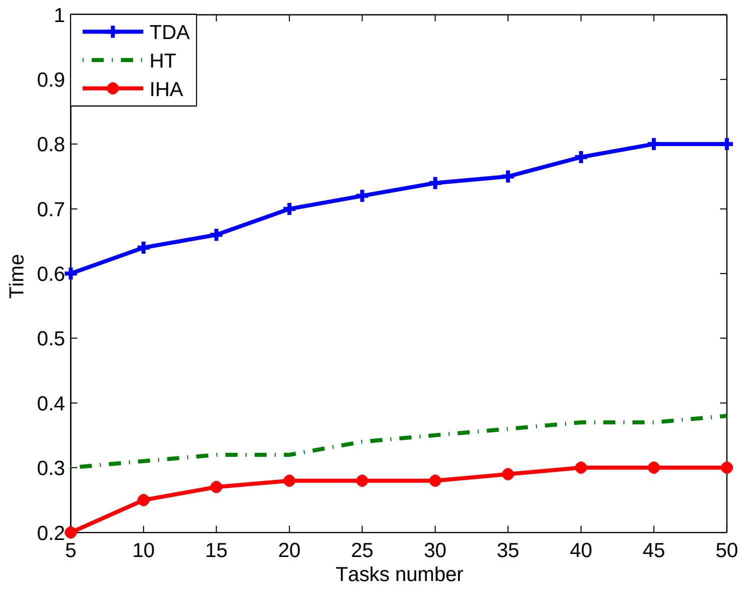

FIGURE 3. Performance at 90 percent system utilization

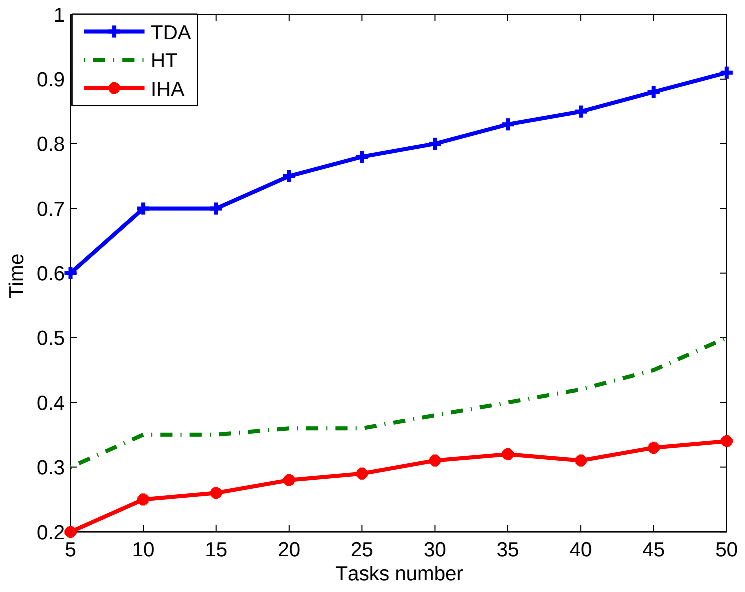

FIGURE 4. Performance at 100 percent system utilization

determined with LL-bound and the later part was analyzed with time demand analysis using lowest priority first approach. Experimental results showed that our technique performed well under various system utilization ranging from 70% to 100% as compared to the existing solutions from the runtime perspective. As future research directions, it will be interesting to study the effect of segregating the task set based on utilization of individual tasks for faster feasibility analysis.

REFERENCES

[1] J. W. S. Liu, *Real Time Systems*, Prentice Hall, 2000.
[2] C. M. Krishna and K. G. Shin, *Real-Time Systems*, McGrawHill, 1997.
[3] H. C. R. Ha and J. W. S. Liu, Experimental analysis of timing validation methods for distributed real-time systems, *The Journal of Supercomputing*, vol.25, no.1, pp.2169-2174, 2003.

[4] S. Alrashed, J. Alhiyafi, A. Shafi and N. Min-Allah, An efficient schedulability condition for non-preemptive real-time systems at common scheduling points, *The Journal of Supercomputing*, vol.72, no.12, pp.4651-4661, 2016.

[5] 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.

[6] J. Orozco, R. Cayssials, J. Santos and R. Santos, On the minimum number of priority levels required for the rate monotonic scheduling of real-time systems, *Proc. of the 10th Euromicro Workshop on Real Time System*, 1998.

[7] S. Alrashed, Reducing power consumption of non-preemptive real-time systems, *The Journal of Supercomputing*, vol.73, no.12, pp.5402-5413, 2017.

[8] L. George, N. Riverre and M. Spuri, Preemptive and non-preemptive real-time uniprocessor scheduling, *Research Report RR-2966*, INRIA, France, 1996.

[9] I. Iimura, 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.

[10] N. Min-Allah and S. U. Khan, A hybrid test for faster feasibility analysis of periodic tasks, *International Journal of Innovative Computing, Information and Control*, vol.7, no.10, pp.5689-5698, 2011.

[11] M. B. Qureshi, M. A. Alqahtani and N. Min-Allah, Grid resource allocation for real-time data-intensive tasks, *IEEE Access*, vol.5, pp.22724-22734, 2017.

[12] E. Bini and G. C. Buttazzo, Schedulability analysis of periodic fixed priority systems, *IEEE Trans. Computers*, vol.53, no.11, pp.1462-1473, 2004.

[13] N. Min-Allah, Y. Wang and J. Xing, Enhanced rate monotonic time demand analysis, *International Journal of Computer Sciences and Engineering Systems*, 2007.

[14] 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.

[15] M. Sjödin and H. Hansson, Improved response-time analysis calculations, *Proc. of the 19th IEEE Real-Time Systems Symposium*, pp.399-409, 1998.

[16] S. Baruah and K. Pruhs, Open problems in real-time scheduling, *Journal of Scheduling*, vol.13, no.6, pp.577-582, 2010.

[17] N. Min-Allah, S. U. Khan, N. Ghani, J. Li, L. Wang and P. Bouvry, A comparative study of rate monotonic schedulability tests, *The Journal of Supercomputing*, vol.59, no.3, pp.1419-1430, 2012.

[18] T. W. Kuo, L. P. Chang, Y. H. Liu and K. J. Lin, Efficient online schedulability tests for real-time systems, *IEEE Trans. Software Engineering*, vol.29, no.8, pp.734-751, 2003.

[19] 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.

[20] N. Min-Allah, S. U. Khan, X. Wang and A. Y. Zomaya, Lowest priority first based feasibility analysis of real-time systems, *Journal of Parallel and Distributed Computing*, vol.73, no.8, pp.1066-1075, 2013.

[21] W. Jia, B. Han, C. Zhang and W. Zhou, Delay control and parallel admission algorithms for real-time anycast flow, *The Journal of Supercomputing*, vol.29, no.2, pp.197-209, 2004.

[22] 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.

[23] N. Min-Allah, S. U. Khan and Y. Wang, Optimal task execution times for periodic tasks using nonlinear constrained optimization, *The Journal of Supercomputing*, vol.59, no.3, pp.1120-1138, 2012.

[24] R. I. Davis, A. Zabos and A. Burns, Efficient exact schedulability tests for fixed priority real-time systems, *IEEE Trans. Computers*, vol.57, pp.1261-1276, 2008.

[25] 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.

[26] 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.

[27] E. Bini, G. C. Buttazzo and G. Buttazzo, Rate monotonic analysis: The hyperbolic bound, *IEEE Trans. Computers*, vol.52, no.7, pp.933-942, 2003.

[28] L. Chen, Y. Lyu, C. Wang, J. Wu, C. Zhang, N. Min-Allah, J. Alhiyafi and Y. Wang, Solving linear optimization over arithmetic constraint formula, *Journal of Global Optimization*, vol.69, no.1, pp.1-34, 2016.