ICIC Express Letters
Part B: Applications
Volume **9**, Number **9**, September **2018**

ICIC International ©2018 ISSN 2185-2766
pp. **907–915**

A FEASIBILITY STUDY OF SORTED TASKS SET

Saleh Alrashed

College of Computer Science and Information Technology (CCSIT)
Imam Abdulrahman Bin Faisal University (IAU)
P.O. Box 1982, Dammam 31441, Saudi Arabia
saalrashed@iau.edu.sa

Received March 2018; accepted June 2018

Abstract. *The feasibility problem of real-time system under static priority assignment has been actively investigated and a plethora of results are available, both in exact and in-exact forms. On the one hand, in-exact conditions belong to pseudo-polynomial complexity class while exact conditions are polynomial. On the other hand, exact conditions can promise up to 100% system utilization while in-exact conditions impose bounds on system utilization. Recently, exact and in-exact conditions were combined for faster feasibility analysis; however, the impact of analyzing the schedulability of maximum tasks with in-exact condition is yet to be studied. To fill this gap, we partition the task set into two parts such that schedulability of maximum number of tasks is determined with the in-exact condition and only a smaller number of tasks are left to be analyzed with exact condition. Our proposed method is based on existing techniques and respects the timing constraints of the presented system as the portioning of task set is only subject to feasibility analysis. Experimental results show that when compared with closely related counterparts, our solution performs better, especially for the task sets having low system utilization.*

Keywords: Operating system, Real-time systems, Fixed-priority scheduling, Feasibility analysis, Deadline monotonic scheduling

1. **Introduction.** Scheduling of real-time systems with deadline guarantees is a very complicated problem due to its instinctive NP nature [1, 2, 3, 4, 5]. The two broader classes of real-time systems are static and dynamic priority systems. In static systems, the priority of task remains the same throughout but priority of a task may change at runtime in dynamic priority systems. These feasibility tests for real-time systems can be divided into in-exact conditions and exact conditions [2, 4, 6, 7, 9, 10, 16, 19].

Various solutions have been proposed to lower the computation cost of feasibility techniques under fixed priority systems and can be divided into exact and in-exact classes [2, 5]. However, the complexity of exact condition is still pseudo-polynomial and attempts have been made to lower the computation cost of such systems [6, 11, 12, 13, 14, 15]. The RM feasibility analysis was extended [18, 20] to many application domains. A Hybrid Approach (HA) was derived in [7] by utilizing both in-exact and exact conditions aiming faster feasibility analysis for Rate Monotonic Scheduling (RMS) [1]. Authors in [7] portioned the task set into two subsets and feasibility was determined accordingly for RMS algorithm where task periods and deadlines have the same values. The need for such solution becomes even more appealing when accuracy is the primary interest while performance is also desirable. In contrast, we propose sorting the task set and extend the results to more general static priority scheduling algorithm known as deadline monotonic [8] by splitting the task set for faster feasibility analysis. Our solution is more appealing to the sorted task list having higher system utilization.

DOI: 10.24507/icicelb.09.09.907

In this work, we study the effect of sorting the task set based on individual tasks utilization and determine its feasibility according to RM scheduling algorithm. We divide a task set such that feasibility of maximum number of tasks is answered by the in-exact condition. This solution enables us to conclude the system feasibility much faster when compared to existing methods that are subject to evaluation under exact test. The feasibility of the first part that consists of sorted task set is tested with LL-bound [1], while the 2nd part is tested with exact condition proposed in [14]. We show that this arrangement significantly reduces the computations cost involved and results are encouraging from the run time perspective.

We divide the rest of the paper into 4 sections. Section 2 introduces the preliminaries and develops the task model. The main work is described in Section 3, while Section 4 presents the experimental results. The paper is concluded in Section 5.

2. **Background and Problem Formulation.** A real-time system consists of a set of periodic tasks and the two main approaches for scheduling such tasks on the processors are (i) preemptive, and (ii) non-preemptive systems. In preemptive scheduling, execution of an executing process may be stopped if a higher priority process requires service, while in non-preemptive systems once a task is scheduled for execution, it runs to completion or until it is blocked. For convenience, the notations that are used frequently in rest of the paper are given in Table 1.

Table 1. Notations

Notation	Meaning
Γ	The set of periodic tasks
c_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	Utilization of Γ
U_i	Utilization of i lower priority tasks
w_i	The cumulative workload of τ_i
$priority(\tau_i)$	Priority of τ_i
P-I	Subset of task set consisting of low utilization tasks
P-II	Subset of task set consisting of high utilization tasks
R_i	Response time of τ_i

For preemptive systems, under static priority assignment, Deadline Monotonic Scheduling (DMS) [8] is optimal in the sense that if there exists a static priority schedule that meets all deadlines for a real-time system, then DMS will produce a feasible schedule. DMS becomes RMS when $d_i = p_i$; $\forall i$, $\tau_i \in \Gamma$.

With DMS, tasks with shorter deadline are assigned higher priorities. In this work, we propose a solution for a faster analysis to determine if a task set can be scheduled with DSM. We assume an implicit deadline model of real-time system $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n periodic tasks, where a task τ_i has computation demand c_i, period p_i, deadline d_i, and response time R_i. Furthermore,

- the number of tasks in the task set is static;
- real time of all tasks are released at $t = 0$;
- all tasks have known worst-case computation times;
- each task instance generated is completed before another jobs of the same task appears;

- there is no dependency among tasks.

The utilization of a task τ_i is $u_i = c_i/\min(d_i, p_i)$ and the total system utilization for i tasks is represented by U_i. The entire task set is DMS feasible if:

$$U_n \leq n\left(2^{1/n} - 1\right). \tag{1}$$

The test given in Inequality (1) is an in-exact condition for DMS with a time complexity of $O(n)$ and called LL-bound. The exact condition for DMS based on R_i was proposed in [13] called Response Time Analysis (RTA). According to RTA, the response time R_i of a task τ_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 of R_i. During the l-th iteration for $l \geq 1$, $R_i^{\#l+1}$ can be computed by

$$R_i^{\#l+1} = c_i + \sum_{j \in T} \left\lceil \frac{R_i^{\#l}}{p_j} \right\rceil c_j, \tag{2}$$

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 1998, authors in [14] proposed an improvement over RTA called Response Time Improved (RTI) by initializing a higher initial guess value to a task τ_{i+} as $R_{i+1}^0 = R_i$, assuming $R_i \leq d_i$. By extending the work done in [7], we now present the main results in Section 3.

3. **Portioned Based Exact-Test.** As seen with Equation (1), all the n tasks are schedulable if the system utilizations are not more than 69%. With our test, we go up to 100% utilization and propose an exact condition that works in hierarchical approach as:

- P-I: represents a group of tasks having lower utilization and schedulability is answered with LL-bound.
- P-II: denotes lower priority tasks and feasibility is determined by RTA.

Now, the feasibility of Γ to be tested with exact condition. To do so, first of all, we sort the task set in the order of utilization. Let us sort the Γ based on ration c_i/d_i. For P-I, we extract task from the Γ such that a generic task τ_i is schedulable as DMS if:

$$U_{i-1} + c_i/d_i \leq i\left(2^{1/i} - 1\right). \tag{3}$$

It is worth noting that, the above formulation does not mean that the lowest utilization task has the highest priority. Rather, P-I is constituted by task having lower utilizations as the motivation for this group is to have maximum task so that only a sub part is left over for P-II. Inequality (3) shows that tasks are grouped into P-I and no task ever misses the deadline. It can be seen in above formulation that tasks are shifted to P-I in their increasing utilization order. Any task τ_j that fails in this test becomes the first element in P-II. We show in the following that no higher utilization task exists in P-I.

Theorem 3.1. *P-I having utilization of* $n\left(2^{(n-1)} - 1\right)$ *is RM-scheduable with LL-bound if all the member tasks in P-I are sorted by low utilization.*

Proof: There are two parts of the proof:

a) To test RM-feasibility of P-I and the proof is given in [1].

b) Determine P-I is sorted by task utilization and we prove this fact by contradiction.

Assuming U represents the total utilization of i lower utilization tasks that are schedulable with LL-bound, except τ_j and τ_k. c_j/d_j and c_k/d_k are the utilization of any two tasks τ_j and τ_k. Let $c_j/d_j < c_k/d_k$ and $\tau_k \in$ P-I while $\tau_j \notin$ P-I. So, when i tasks are schedulable, then:

$$U \leq i\left(2^{(1/i)} - 1\right) \tag{4}$$

Since i tasks are schedulable, Inequality (4) holds. Adding τ_k:

$$U + c_k/d_k \leq i\big(k^{1/k} - 1\big)$$

Adding τ_j:

$$U + c_j/d_j \leq i\big(j^{1/j} - 1\big)$$

Therefore,

$$c_k/d_k \leq c_j/d_j$$

which is contradiction. This completes the proof. □

It is worth noting that there is no restriction that all tasks in P-I are higher priory tasks as sorting is done based on the utilization factors and not on task priorities while grouping them. The complexity of P-I is due to two factors (i) sorting tasks, and (ii) checking against LL-bound, which is $O(n \log n)$ and $O(n)$, respectively.

We now populate the second subset P-II. It is evident from Inequality (3) that P-II consists of only higher utilization tasks and it is unknown weather it is DMS feasible or not. Since in-exact conditions are not applicable to P-II, we apply RTA for answering the feasibility of P-II. Before that we relax the assumption of sorted task needed for P-I. For P-II, all tasks are re-arranged by the task deadlines, i.e., the larger the task deadline is, the lower the priority is. Assuming τ_k is the last task that was assigned to subset P-I and task τ_l is not feasible in P-I, so τ_l becomes a candidate for subset P-II. A task cannot be spitted between two groups and each task must be either in group P-I or P-II. For any two tasks τ_i and τ_j, the corresponding sets are represented by P-I and P-II, respectively. If $\tau_i \in$ P-I, then the τ_i does not exist in P-II.

Now, the schedulability of P-II has to be analyzed with RTI. The actual priorities of the task in P-II are assigned as per standard DMS policy. These tasks are the tasks with higher individual utilizations. It is very likely that majority of tasks in P-II have larger task periods than those in P-I. The associated advantage with larger periods is that R_i makes significant impact in the analysis for lower priority tasks $\{\tau_{i+1}, \tau_{i+2}, \ldots, \tau_n\}$. Since we cannot ignore the task in P-I, the workload due to these tasks must be considered while testing the DM schedulability of τ_k. For each task τ_l in P-II, RTI is used to determine its feasibly. The task continues as long as $R_j <= d_j$; otherwise, P-II is declared infeasible. With this formulation, it can be seen that the task parameters remain intact as per original set. Once feasibility of P-II is answered positively, the actual priority of the task assigned as per standard DMS policy is applied. As a concluding step, P-I and P-II are combined and schedule the Γ according to DMS on a uni-processor systems. These intermediate arrangements pertaining the feasibility analysis only and the spirit of DMS algorithm is respected. As a concluding step, we assign static priorities on task deadlines such that for any two tasks τ_i and τ_j, priority$(\tau_i) >$ priority$(\tau_j) \Rightarrow$ deadline$(\tau_i) <$ deadline(τ_j), while ties are broken arbitrarily. We represent our technique by Sorted-Technique (ST) and sketch the details in Algorithm 1 that accepts Γ and returns whether Γ is DMS schedulable or not.

4. **Experimental Results.** In this section, we compare ST with two closely related feasibility tests: RTI, HA. We study the behavior of these tests from the computation cost perspective. It should be noted that both RTI and HA are exact conditions and may answer schedulability of Γ when total system utilization is less then or equal to 100%. For experiments, to align with existing works, we generate random task sets from size 10 to 50. The task size is increased with step size of 10. Task deadlines are produced in the range of $[10, 10000]$ in Matlab with uniform distribution. Similarly, for corresponding task computations, random values are taken in the range of $[1, d_i]$. Experimentation is done on Pentium-VI (Intel, Core-2 CPU), 1.4 GHz with 2GB RAM. Each reading is taken from running a task set for 200 iterations. We normalize all the values to have a better idea of the performance of the related techniques.

Algorithm 1.

procedure ST(Γ)
sort Γ in descending order from task utilization perspective;
for all $\tau_i \in \Gamma$ **do**
  if $U_{i-1} + c_i/d_i \leq i\left(2^{1/i} - 1\right)$ **then**
    τ_i is DM schedulable;
    P-I += τ_i;
  end if
  if $\tau_i == \tau_n$ **then**
    Γ is DM schedulable;
    EXIT;
  end if
end for
P-II = (Γ − (P-I) − τ_i);
sort P-II in descending order of task deadline;
for all $\tau_j \in$ P-II **do**
  $R_j^{\#l+1} = c_j + \sum_{k \in \text{P-I}} \left\lceil \frac{R_i^{\#l}}{p_j} \right\rceil c_j$;
  calculate $R_i^{\#l+1} = R_i^{\#l}$;
  if $R_j^{\#l+1} > d_j$ **then**
    Γ is not DM schedulable;
    EXIT;
  else
    τ_j is DM schedulable;
    P-II += τ_j;
  end if
  if $\tau_j == \tau_n$ **then**
    P-II is DM schedulable;
    Γ = P-I + P-II;
    sort Γ per DM priority;
    Γ is DMS feasible;
    EXIT;
  end if
end for
end procedure

From Figure 1, it can be seen that the values for ST are low and almost constant. The behavior is understandable as maximum tasks belong to P-I due to 75% utilization and thus schedulable with LL-bound. Only few tasks are now in P-II when the task set consists of 10 tasks and the number increases when the task set size increases but again its insignificant increase. On the other hand, HA is better than RTI as a portion of the task set is analyzed with LL-bound in HA while all tasks are tested with RTI, which is pseudo-polynomial in nature. The same trends continue in Figures 2-4. For our experiments, Figures 5 and 6 present the maximum load to RTI, HA and ST. Under 95% and 100% utilization, more tasks will be placed in P-II with ST and the same for HA, and hence all techniques take more time as compared to 75% system utilization. Interestingly, the values in Figure 5 are on lower side, especially for RTI. All the techniques answer the DM feasibility much faster in Figure 6 when CPU demand is 100%. The reason for these lower values is that DMS is only optimal for static priority systems and fails when system utilization increases. For higher utilizations, dynamic priority systems become more appealing and promise schedulability of the Γ as long as utilization is less than

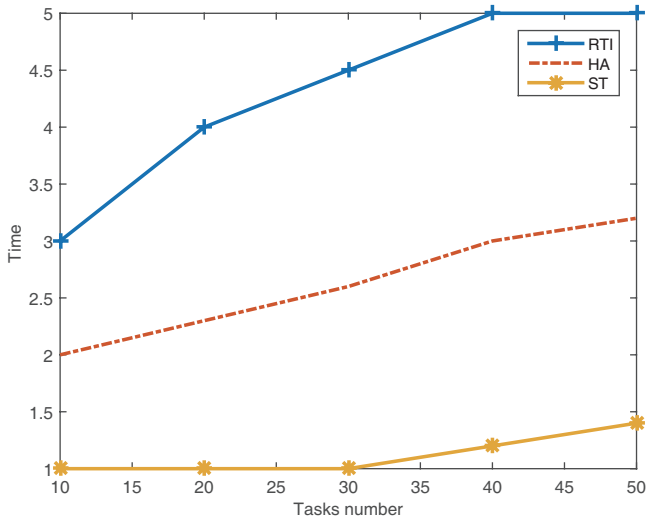

FIGURE 1. Performance at 75 percent system utilization

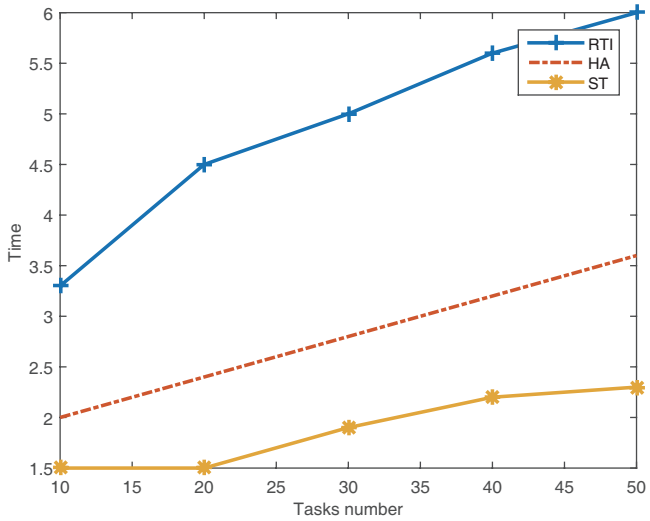

FIGURE 2. Performance at 80 percent system utilization

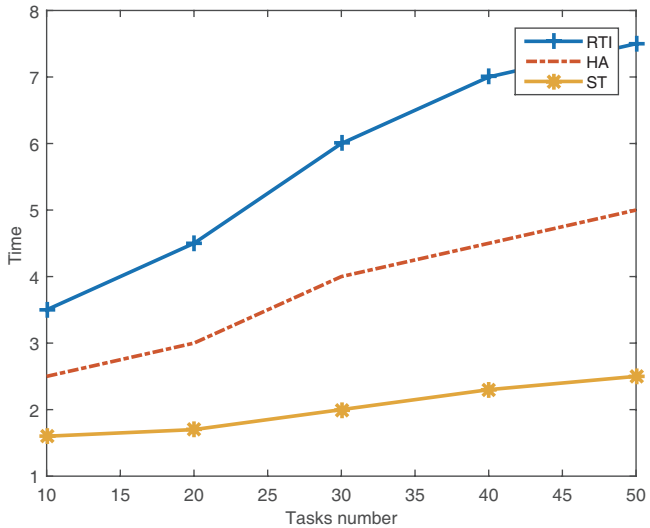

FIGURE 3. Performance at 85 percent system utilization

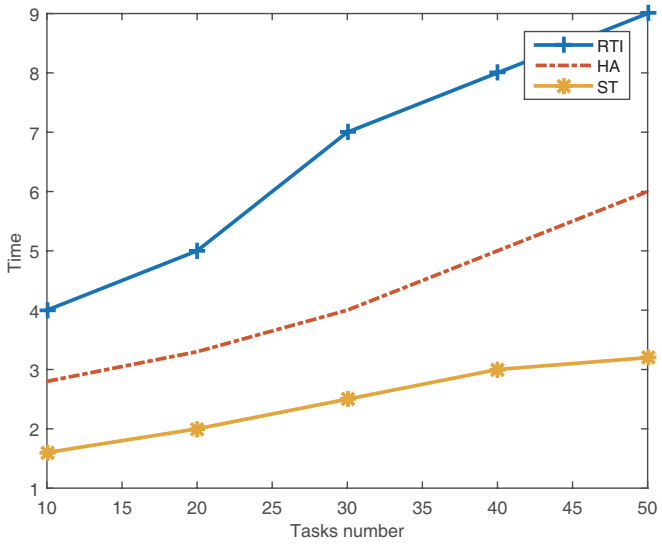

FIGURE 4. Performance at 90 percent system utilization

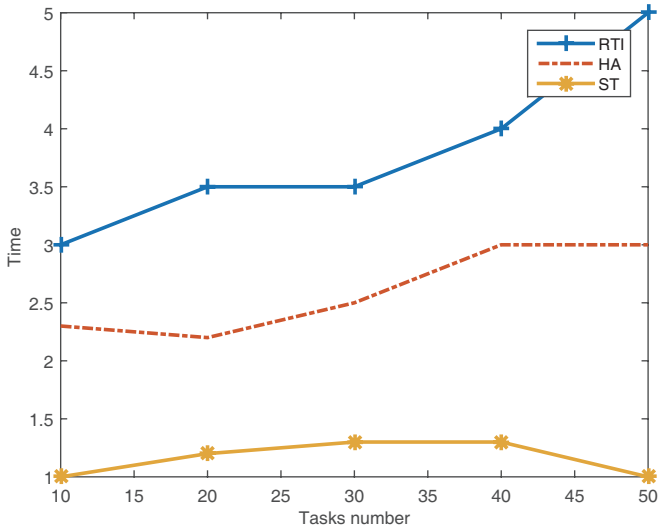

FIGURE 5. Performance at 95 percent system utilization

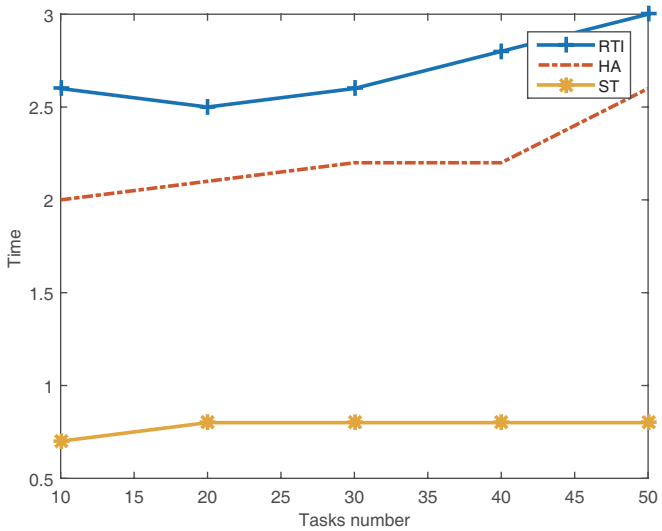

FIGURE 6. Performance at 100 percent system utilization

100%, but those techniques are out of the scope of this work. With higher utilization, the task system is very likely to be infeasible on single CPU. In Figure 6, the values go even lower. Under 100%, ST and HA are better as some portion is tested with LL-bound while few tasks are analyzed by exact condition, while RTA has to test all task sets with exact condition starting from τ_1 and so on in decreasing priority order. ST is better in the sense that a good portion of tasks set is analyzed with LL-bound and it is very likely that one few tasks are analyzed before the system is declared infeasible as per DM scheduling algorithm.

5. **Conclusion.** In this work, we kept the timing constraints of the system intact while dividing the system into two parts for faster feasibility analysis. We divided the real-time system into two subsets such that the feasibility of the larger part is analyzed with in-exact test. The first subset was sorted in ascending order of task utilization for accommodation maximum tasks. The other subset was tested with exact test. After analyzing the feasibility of the task set, the deadline monotonic priority assignment was used for scheduling the system on a uni-processor system. As a future research direction, the effect of task set size as well as the role of higher guess values will be interesting to research community.

REFERENCES

[1] 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.

[2] J. W. S. Liu, *Real Time Systems*, Prentice Hall, 2000.

[3] 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.

[4] C. M. Krishna and K. G. Shin, *Real-Time Systems*, McGrawHill, 1997.

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

[6] S. Alrashed, An improved hybrid test for feasibility analysis of periodic tasks, *ICIC Express Letters*, vol.12, no.8, pp.759-766, 2018.

[7] 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.

[8] J. Y.-T. Leung and J. Whitehead, On the complexity of fixed-priority scheduling of periodic, real-time tasks, *Performance Evaluation*, vol.2, no.4, pp.237-250, 1982.

[9] 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.

[10] 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.

[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] 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.

[13] 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.

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

[15] N. Min-Allah, S. U. Khan and Y. Wang, Optimal task execution times for periodic tasks using nonlinear constrained optimization, *The Journal of Supercomputing*, pp.1-19, 2010.

[16] 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.

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

[18] 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.69-102, 2017.

[19] 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.

[20] Y. Lyu, L. Chen, C. Zhang, D. Qu, N. Min-Allah and Y. Wang, An interleaved depth-first search method for the linear optimization problem with disjunctive constraints, *Journal of Global Optimization*, vol.70, no.4, pp.737-756, 2018.