Journal of Computer Science

Original Research Paper

Integrating Lowest Priority Approach with Largest Point Scheme for Faster Feasibility Analysis

Nasro Min-Allah

Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam, Saudi Arabia

Article history
Received: 13-03-2019
Revised: 16-04-2019
Accepted: 23-04-2019

Email: nabdullatief@iau.edu.sa

Abstract: Recently many solutions have been proposed to lower the computational cost of feasibility analysis for real-time systems. The computational cost of feasibility tests can be lowered by strategies such as lowering the number of scheduling points needed during analysis, starting feasibility analysis from lowest priority, or starting schedulability tests for a task with larger scheduling point. All these techniques significantly reduce the computation time of feasibility analysis for fixed priority systems. The computation time of such tests can be further reduced by combining various solutions for efficient feasibility analysis of periodic task sets. In this work, we integrate both lowest priority first with largest points first solution to derive a faster feasibility analysis test for fixed priority system. Our experimental evaluations suggest that the proposed technique significantly lowers the computational cost of the test when system utilization is in the range of 80% or when the ratio between the task period of a lower priority task and the highest priority task is large.

Keyword: Real-Time Systems, Operating System, Fixed-Priority Scheduling, Feasibility Analysis

Introduction

One of the main component of an operating system is multitasking that enables running multiple tasks on the same computer system. In multitasking systems, a scheduler run process tasks based on some criteria such first come first out, shortest job first or round robin etc. All these solutions have pros and cons and no single scheduling policy is applicable to a diverse set of applications (Liu and Layland, 1973; Leung and Whitehead, 1982; George *et al*., 1996; Min-Allah, 2019; Bini and Buttazzo, 2004; 2001). These techniques are good for general tasks where the motivation is full utilization of re- sources or achieving higher throughput. However, aforementioned techniques can not be used of real-time system due to specific nature of tasks where timing constrains must be guaranteed under any possible circumstances.

In operating systems, scheduling can be classified into two main types of preemptive and non-preemptive classes. Under preemptive class, an executing task is preempted whenever another task with higher priority arrives while non-preemptive policy puts no such restriction and let the executing task to its completion. Form system utilization perspective, preemptive scheduling is preferred over non-preemptive counterpart. Various scheduling techniques have been proposed for real-time system (Liu and Layland, 1973; Leung and Whitehead, 1982; George *et al*., 1996; Katcher *et al*., 1993; Lehoczky *et al*., 1989; Bini and Buttazzo, 2001; Han and Tyan, 1997; Kuo *et al*., 2003; Audsley *et al*., 1993; Sjodin and Hansson, 1998) that ensures the timing requirements are met by prioritizing task executions running on the system. For instance, Rate Monotonic Scheduling (RMS) (Liu and Layland, 1973) strategy assigns priority by task activation rate while Deadline Monotonic Scheduling (DMS) (Leung and Whitehead, 1982) algorithm assigns priorities based on tasks deadlines. Both RMS and DMS are static priority assignment algorithm due to its static priority allocation to individual tasks which never changes at run time. The main limitation of such techniques is poor CPU utilization, especially with existing inexact conditions. To encounter this limitation, a dynamic scheduling algorithm known as Earliest Deadline First (EDF) was also derived in (Liu and Layland, 1973) where tasks priorities are given from the perspective of deadlines. With EDF, there is no need to define off-line priorities as

it keeps changing at run time. The closer is the deadline, the higher is a task priority and so on. When it comes to predictability, static scheduling are more predictable due to its fixed priority assignment which can easily identify that which particular periodic task will miss the deadline when the system becomes overloaded.

To determine, weather a given real-time system meets its associated timing constraints, feasibility analysis is a must for any intended scheduling algorithm to be used for scheduling periodic tasks. For RMS and DMS, there exist two types of feasibility analysis tests, i-Inexact and ii-Exact condition. Inexact-conditions (Liu and Layland, 1973; Kuo and Mok, 1991; Bini and Buttazzo, 2001) are very fast while exact- conditions (Lehoczky *et al*., 1989; Alrashed, 2018; Audsley *et al*., 1993; Bini and Buttazzo, 2004) are slow from implementation point of view. On the contrary, exact-conditions result in better system utilization as compared to inexact conditions. Such systems were also studied from the perspective of liner optation problem using constraint formula and disjunctive constraints (Chen *et al*., 2017; Lyu *et al*., 2018). In this paper, we extend the work done in (Min-Allah, 2019) with lowest priority first approach for preemptive scheduling. Though there exist feasibility analysis techniques (Audsley *et al*., 1993; Sjodin and Hansson, 1998) that are superior to the scheduling points alternatives but those solutions are based on analysis of task response time and thus out of the scope of this paper. This work aims to integrate two recently developed approaches namely lowest priority first approach (Min-Allah *et al*., 2013) and largest point first (Min-Allah, 2019) to study its impact on the schedulability analysis of periodic task set.

Recently, various solutions were presented to lower the computation cost of exact- conditions (Alrashed, 2018; Min-Allah and Khan, 2011; Bini and Buttazzo, 2004; Joseph and Pandya, 1986; Leung and Whitehead, 1982; Min-Allah, 2019; Alrashed *et al*., 2016; Min-Allah *et al*., 2013; Sjodin and Hansson, 1998). Feasibility study was discussed in (Alrashed, 2018) by sorting tasks set while a hybrid technique was presented in (Min-Allah and Khan, 2011). Scheduling points were restricted to a subset in (Bini and Buttazzo, 2004). Similarly, feasibility was determined with lowest priority first approach in (Min-Allah *et al*., 2013). A hybrid test was established in (Min-Allah and Khan, 2011) by using both exact and inexact conditions. Recently, the feasibility of a task set was analyzed by using a largest point in the set of candidate scheduling points in (Min-Allah, 2019). In this work, we integrate the lowest priority first fashion at task set level and use the largest point strategy when schedulability is concerned with a single task. Our technique can be classified under exact class of feasibility analysis as the complexity is pseudo-polynomial. Our experimental results show significant improvement in run time when compared to related solutions.

To maintain a good flow in the work, we divide the paper into 4 Sections. Section 2 discusses related work and the system model to be used for establishing the test. Details of our improved test are given in Section 3 while experimental results are shown in Section 4. We highlight conclusion and potential future research directions in Section 5.

Background Work and System Model

In this work, we assume a real time systems which is a collection of independent periodic tasks. A periodic task τ_i in the task set is represented by the three essential parameters c_i, p_i and d_i. Parameter c_i represents the CPU execution time needed for an instance (job) of a task before its next instance arrives. Similarly, p_i denotes the task period where jobs of τ_i are released periodically after p_i intervals, while d_i shows the task deadline. For the successful completion, each instance must receive c_i units of CPU slots before its respective deadline d_i. We assume the first jobs of each task is released at $t = 0$. The scheduling algorithm used is RMS and task set consist of n tasks, while the underlying system has a single processor system.

To answer the feasibility of the aforementioned task model with RMS on uni-processor system, feasibility tests are performed and many tests are available in real-time systems literature. As discussed in Section 1, feasibility tests are of two types i.e., inexact and exact tests. In this context inexact test is a sufficient condition while exact test is both necessary and sufficient conditions. The first inexact condition for RMS was derived in (Liu and Layland, 1973) as: A periodic task system of independent tasks is RMS feasible if:

$$\sum_{i=1}^{n} \frac{c_i}{p_i} \leq n\left(2^{1/n} - 1\right) \tag{1}$$

In the above expression, it can be seen that Inequality 1 puts a bound on system utilization of $ln(n)$ when n approaches ∞. Since the introduction of preemptive real-time systems scheduling theory and corresponding feasibility condition in 1973 (Liu and Layland, 1973) for periodic tasks under a simple periodic task set, many solutions have been presented by relaxing the limitations of the task set such as making task periods harmonic etc. Extending the work done in (Liu and Layland, 1973), authors in (Bini and Buttazzo, 2001) derived an inexact condition with higher acceptance ratio: A periodic task system of independent tasks is RMS feasible if:

$$\prod_{i=1}^{n}\left(1 + \frac{c_i}{p_i}\right) \leq 2 \tag{2}$$

Both Inequality 1 and 2 are general condition under inexact class and hence determine the RMS feasibility of

the system when feasibility conditions are true, however nothing can be said when conditions are false. It can be noted that full utilization of the system can not be fully achieved with inexact conditions. To have a higher system utilization more tests were proposed in (Kuo and Mok, 1991; Katcher *et al*., 1993; Han and Tyan, 1997). For instance, to achieve up to 100%, authors in (Kuo and Mok, 1991) tuned task periods to harmonic which is a restricted case but the test is of polynomial complexity and can be used in online systems.

On the other hand, exact conditions are both necessary and sufficient conditions (Lehoczky *et al*., 1989; Min-Allah and Khan, 2011; Alrashed, 2018; Sjodin and Hansson, 1998; Joseph and Pandya, 1986; Audsley *et al*., 1993; Bini and Buttazzo, 2004) and can result in higher utilization but the complexity associated is pseudo-polynomial. These solution test feasibility of each task one by one answer system feasibility as true or false. The role of workload is of primary interest to this class and has been discussed in related literature (Khan and Min-Allah, 2012; Min-Allah *et al*., 2012; Kolodziej *et al*., 2011; Min-Allah *et al*., 2010).

Task τ_i gets the desired CPU time at any time t, when a job from the higher priority task in the system, in addition to the computation time of the task τ_i, is assigned CPU time at or before the time t. For simplicity, time can be assumed as an integer number and computation time of a task depends on the system speed i.e., a task that takes 2 ms on a CPU having 2 GHz speed can tentatively take 2 ms when running at 1 GHz. It is worth noting that task periods are equal to task deadlines in implicit deadline model which is applicable in this work. For testing RM schedulability of an individual task τ_i at a time t, the cumulative execution demand is constituted by c_i as well as the total CPU demand of all the higher priority periodic tasks starting from τ_{i-1} up to τ_1. This is due to the fact that the processor can only be given to a low priority task when there does not exist any high priority task. Therefore, the maximum workload on CPU at time t can be written as:

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

In a periodic task set, a task τ_i is schedulable in the time interval $[0, p_i]$ when:

$$l_i = \min_{0<t\le p_i} \left(w_i(t) \le t \right) \tag{4}$$

The above inequality shows that a task τ_i fulfills the required execution requirements at or before time $t \in [0, p_i]$, iff the entire request from all i-1 higher priority tasks and computation time of τ_i, is provided at t. The problem is that t is a continuous variable and there exist infinite numbers of candidate scheduling points to be tested for a task τ_i.

Entire system τ is RMS feasible iff:

$$L = \max_{1<i\le n} \left\{ \min_{0<t\le p_i} \frac{w_i(t)}{t} \right\} \le 1 \tag{5}$$

Lehoczky *et al*. (1989), provided a solution where in finite number of points in the time interval $[0, p_i]$ was rationally restricted to a set of candidate points where computational load changes due to arrival of high priority tasks. Lehoczky *et al*. (1989) showed that $w_i(t)$ remains constant, except at finite number of points, where tasks are released, called RMS scheduling points. The aforementioned work, ignored the task periods of tasks that have priority lower than tau_i as those tasks can not be allocated CPU slots as long as τ_i or any higher priority tasks needs CPU time. In this paper, we use schedulability and feasibility inter-changeable.

Let:

$$S_i = \left\{ ap_l \mid l = 1,...,i; a = 1,...,\left\lfloor p_i / p_l \right\rfloor \right\} \tag{6}$$

For checking schedulability of a single task, Lehoczky *et al*. (1989) determined if an individual task τ_i is schedulable with RMS:

Theorem 2.1. (Lehoczky et al., 1989)

Given a set of n periodic tasks $\tau_1,\ldots, \tau_n$, τ_i can be feasibly scheduled for all tasks phasings using RM iff:

$$l_i = \min_{t\in S_i} \frac{w_i(t)}{t} \le 1 \tag{7}$$

The periodic task set τ is RMS schedulable on a single CPU system iff:

$$L = \max_{1\le i\le n} \left\{ \min_{0<t\le p_i} \frac{w_i(t)}{t} \right\} \le 1 \tag{8}$$

It is clear from the above set that all elements in the set of scheduling point sets are task periods of higher priority tasks while low priority task periods do not contribute any point to the said set. Having such set of potential points that is constituted by task periods, RMS schedulability of τ_i is checked by:

Theorem 2.2. (Lehoczky et al., 1989)

A periodic tasks set $\tau_1,\ldots, \tau_n$, τ_i can be feasibly scheduled on a uni-processor system using fixed priority scheduling algorithm for all tasks phasings iff:

$$l_i = \min_{t \in S_i} \frac{w_i(t)}{t} \leq 1 \quad (9)$$

Many authors extended the work done in Lehoczky *et al*. (1989) by proposing feasibility test with lower computational cost.

Improved RMS Feasibility Test

Traditionally, highest priority first approach is used to determine feasibility of the task set and starts testing schedulability with highest priority task in the task set. This policy continues till the lowest priority task is determined RM schedulability, which answers the RMS schedulability of the entire task set positively. Recently, an interesting approach was explored in (Min-Allah *et al*., 2013) that check system feasibility from the perspective of lowest priority first approach with:

Theorem 3.1. (Min-Allah et al., 2013)

A periodic task set τ is always RM schedulable if the lowest priority task τ_n is schedulable at some point $t \in \Upsilon$ such that $\Upsilon = \bigcap S_i \neq \phi$, $\forall i$, $1 \leq i \leq n$.

This observation is justified as it is very likely that if a system in infeasible it is due to lower priority task in case of RMS. Similarly, Min-Allah (2019) discussed that it is a rational approach to check RMS schedulability of a task at higher points.

Corollary 3.1. (Min-Allah, 2019)

A system consisting of n periodic tasks given in the descending priority order τ_1, τ_2…, τ_n, τ_i can be feasibly scheduled for all tasks phasings using RM iff:

$$\max_{1 \leq i \leq n} \left\{ \min_{t \in O_i'} \frac{w_i(t)}{t} \right\} \leq 1 \quad (10)$$

where, O_i' is a set of scheduling points in descending order and hence the first element is the largest task period. We show that by combining both Theorem 3.1 and Corollary 3.1, the computation cost can be lowered significantly. First of all, we order the entire task with descending order so that the first half element represents the lowest priority tasks and the last element denotes the task with smallest period and hence highest priority is as-signed to this task. As we know, the task priorities and periods are inversely proportional, a higher priority task has to be completed in a smaller time window as compared to lower priority tasks. In preemptive scheduling, a lower priority task can be preempted multiple times in the interest of a higher priority task. This is the reason that preemptive scheduling under RMS promises higher system utilization. On the contrary, non-preemptive scheduling is straight forward and easy to code. We ignore the preemption cost in this work. In next step, we obtain a set of candidate scheduling points denoted by O_i' and start feasibility of a task τ_i with largest scheduling points. Such arrangements help in lowering the computational cost of the overall system. The values of task periods also inuence feasibility analysis at task level and the same is true when the computational cost of individual tasks are very small. We first sort the task set in reverse order and then check feasibility with lowest priority task first and so on. Again this solution can answer task set infeasibility much early as the lowest priority tasks is generally the one which is always unschedulable with RMS when the task set is infeasible. This combination results in lowering the computational cost of the feasibility tests under RM scheduling on single CPU system. We represent our work in the following corollary:

Corollary 3.2.

Given a set of *n* periodic tasks in ascending priority order τ_n, τ_{n-1}…, τ_1, τ_i, can be feasibly scheduled for all tasks phasings using RMS iff:

$$\forall (i : n,...,1) \left\{ \min_{t \in O_i'} \frac{w_i(t)}{t} \right\} \leq 1 \quad (11)$$

Proof

It can be directly followed by Theorem 3.1 and Corollary 3.1.

It can be seen from Corollary 3.2 that parameters of the task set are intact while feasibility is tested in lowest priority fashion. On the task level, RMS schedulability is determined at scheduling points O_i' in a set where the first element is task period of the task τ_i. The entire task set is RMS schedulability when all the tasks in the task set are declared schedulable. The original task set has been arranged in such a way that the last element denotes the task with highest priority which is always schedulable. Our technique work efficiently for both cases when system utilization is low or when the task set becomes infeasible with presence of unschedulable tasks. With low utilization, our test find schedulability of a task at higher points and hence converges early while in case of overloaded system, the integration of lowest priority first approach determines system infeasibility much faster. This approach is exact condition for RMS feasibility analysis and when a task set feasibility is determined by any exact condition, Corollary 3.2 never fails.

Experimental Results

We now show the experimental evaluation of Corollary 3.2 and compare our results with previous techniques. In our experiments, we use uniform distribution for obtaining task execution requirements and

task periods. A similar analysis is done in (Min-Allah, 2019) but the focus in that work was on highest priority first while we study in feasibility problem using lowest priority first counterpart. Detailed aspects of our analysis an be made by using the scientifically model (Gonzlez-Briones *et al*., 2018) but to align with previous literature, we follow the approach used in (Min-Allah, 2019). We run each experiment 300 times and plot the average values in Fig. 1. First we calculated task period p_i and then c_i were obtained in range [1, p_i]. We compare our results with Theorem 3.1 and the approach used in Corollary 3.1 as these are closely related techniques. We use a sample size of 10-100 where we generate task set starting from 10 tasks and then increase the size by adding 10 more tasks. We plot normalized values for the execution time of each test under various system utilization. We represent Theorem 3.1 by Lowest Approach (LA), Corollary 3.1 by Largest Point (LP) and Corollary 3.2 by Lowest Approach with Largest Point (LALP).

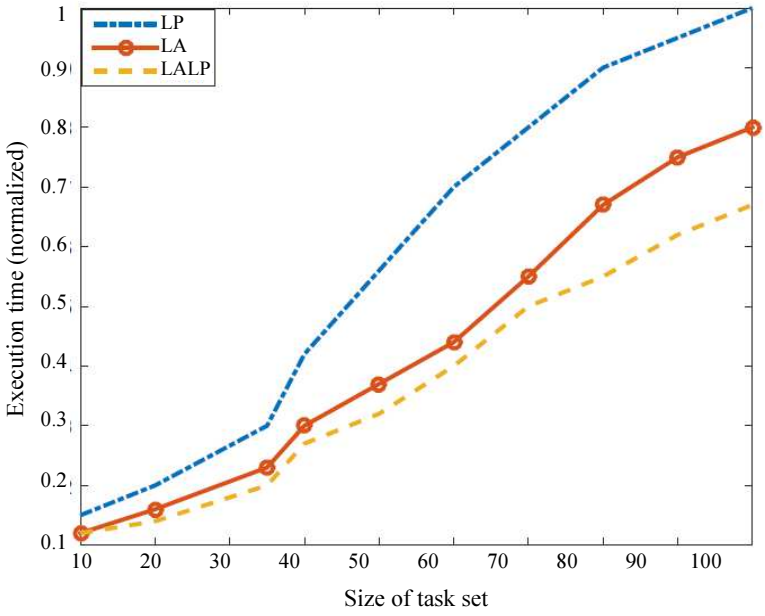

Fig. 1: Run times at 80% system utilization when $P_n = P_1 = 1000$

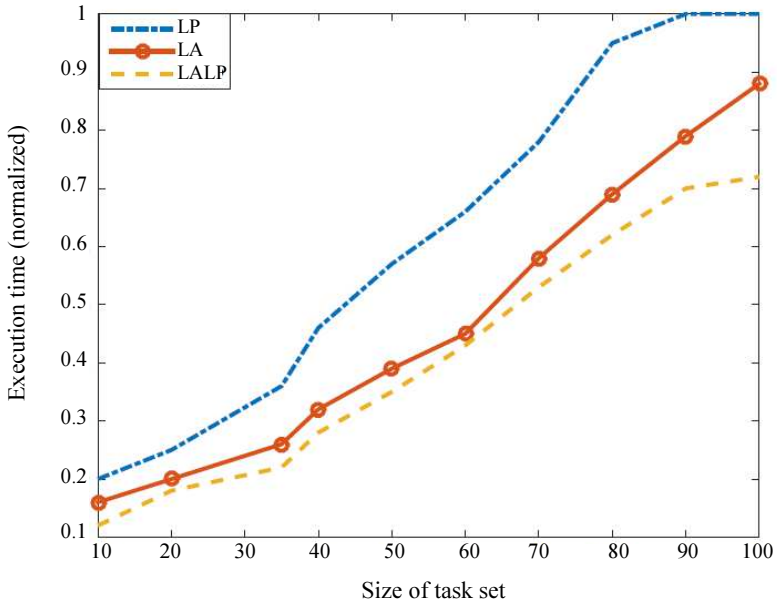

Fig. 2: Run times at 90% system utilization when $P_n = P_1 = 1000$

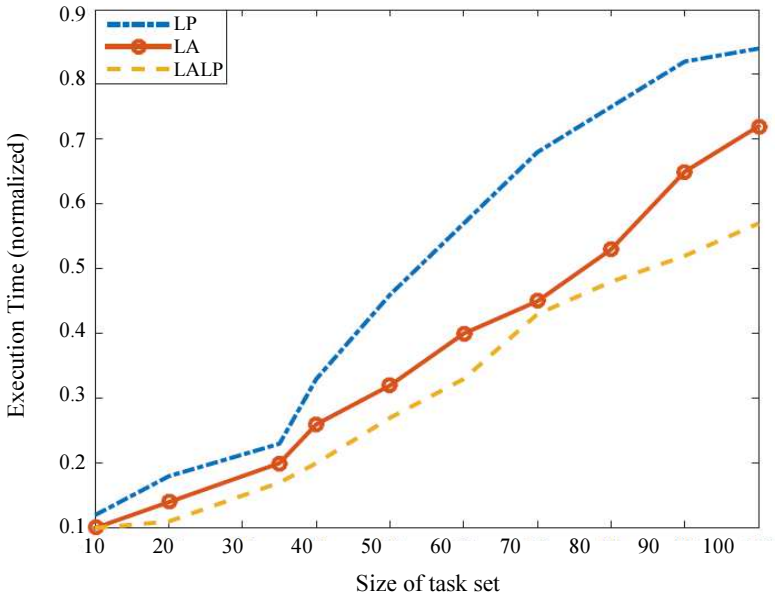

Fig. 3: Run times at 80% system utilization when $P_n = P_1 = 100000$

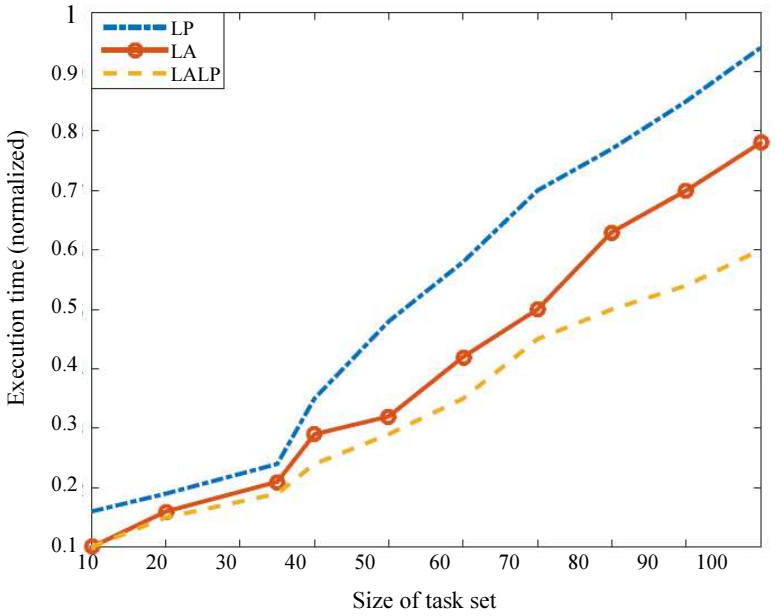

Fig. 4: Run times at 90% system utilization when $P_n = P_1 = 100000$

In our first experiments, we kept the keep the ratio of shortest and largest task percoid as 1000 in Fig. 1 and 2 and then the ratio was increased to 100000 in Fig. 3 and 4. For each figure, we assigned system utilization as 80% against Fig. 1 and 3 and also 90% utilization for Fig. 2 and 4. Below 70% it is understood that the task set if RMS feasible and hence we test our technique under appropriate system utilization. It can be see that when utilization is low in Fig. 1, the LA is almost similar to LP.

While LALP is superior to both LA and LP. This behaviour is due to the fact that all the task are schedulable at such lower utilization and hence both LP and LA are going to check feasibility of all points. Similarly, the performance of all techniques degrades at higher utilization of 90% in Fig. 2. On the other hand, all techniques are promising when system utilization is 80% in Fig. 3 and the values for task periods are high. This situation is due to the fact that more the probability of missing deadline by lower priority tasks is very low. The runtime will become even lower for LP and LALP when utilization is increased but this is not due to the feasibility of the task set rather the task set will become

infeasible and hence both LP and LALP will terminate early. LALP outclasses both counterparts in Fig. 3 as only few points are needed to test system RMS feasibility with such large task periods. The results obtained for lower utilization and higher period ration are promising for LALP. The same trend continues in Fig. 4 but utilization now inuences the run time of feasibility tests as the cumulative demand is higher at utilization of 90% and many lower priority task can miss the deadline. This situation is understandable as task infeasibility is determined much early with LA and LALP as compared to LP. The improvement due to LALP is due to lesser number of inequalities to be tested with Corollary 3.2 which is the main contribution of this work.

Conclusion and Future Work

We exploited i-lowest priority first approach at the task set level and ii-largest point technique at the task level for an improved feasibility test. This combination lowered the computational cost of the RM feasibility test. Two existing solutions were combined to obtain an efficient feasibility test that determines rate monotonic schedulability of the task set on a uni-processor system. Feasibility of tasks were checked with lowest priority first approach and for an individual task, schedulability was analyzed by starting with larger points and so on. Our experimental results showed that proposed technique significantly lowered the computational cost as compared to existing alternatives. Our technique has the advantage to check system feasibility much faster in general and in particular for RMS infeasible task sets on the single processor system. As a future work, it will be interesting to use hybrid approach for answering a subset of tasks with an inexact condition and check schedulability of the remaining tasks with the solution sketched in this paper.

Acknowledgment

The author would like to extend sincere thanks to the anonymous referees and colleagues at the College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, for their valuable suggestions while revising this paper.

Author's Contributions

A feasibility test is obtained by combining the largest point approach at the task level while starting feasibility of the overall task set with lowest priority.

The solution concludes RMS feasibility at lowered computational cost as compared to existing counterparts. The test suggests promising results when system utilization is low or when the ratio between any two tasks periods is high.

Ethics

The experimental and theoretical results reported in work were obtained keeping in view the standard ethics practices employed in scientific research.

References

Alrashed, S., 2018. An improved hybrid test for feasibility analysis of periodic tasks. ICIC Express Lett., 12: 759-766. DOI: 10.24507/icicel.12.08.759

Alrashed, S., J. Alhiyafi, A. Shafi and N. Min-Allah, 2016. An efficient schedulability condition for non-preemptive real-time systems at common scheduling points. J. Super Comput., 72: 4651-4661. DOI: 10.1007/s11227-016-1751-6

Audsley, N.C., A. Burns, K. Tindell and A. Wellings, 1993. Applying new scheduling theory to static priority preemptive scheduling. Software Eng. J., 8: 284-292. DOI: 10.1049/sej.1993.0034

Bini, E. and G. Buttazzo, 2001. A hyperbolic bound for the rate monotonic algorithm. Proceedings of the 13th Euromicro Conference on Real-Time Systems, Jun. 13-15, IEEE Xplore Press, Delft, The Netherlands, pp: 59-66. DOI: 10.1109/EMRTS.2001.934000

Bini, E. and G.C. Buttazzo, 2004. Schedulability analysis of periodic fixed priority systems. IEEE Trans. Comput., 53: 1462-1473. DOI: 10.1109/TC.2004.103

Chen, L., Y. Lyu, C. Wang, J. Wu and C. Zhang *et al*., 2017. Solving linear optimization over arithmetic constraint formula. J. Global Optimiz., 69: 69-102. DOI: 10.1007/s10898-017-0499-8

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

Gonzlez-Briones, A., J. Prieto, F. De La Prieta, E. Herrera-Viedma and J. Corchado, 2018. Energy optimization using a case-based reasoning strategy. Sensors, 18: 865-865. DOI: 10.3390/s18030865

Han, C.C. and H.Y. Tyan, 1997. A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms. Proceedings of the 18th IEEE Real-Time Systems Symposium, Dec. 2-5, IEEE Xplore Press, San Francisco, CA, USA, pp: 36-45. DOI: 10.1109/REAL.1997.641267

Joseph, M. and P. Pandya, 1986. Finding response times in a real-time system. Comput. J., 29: 390-395. DOI: 10.1093/comjnl/29.5.390

Katcher, D.I., H. Arakawa and J.K. Strosnider, 1993. Engineering and analysis of fixed priority schedulers. IEEE Trans. Software Eng., 19: 920-934. DOI: 10.1109/32.241774

Khan, S.U. and N. Min-Allah, 2012. A goal programming based energy efficient resource allocation in data centers. J. Supercomput., 61: 502-519. DOI: 10.1007/s11227-011-0611-7

Kolodziej, J., S.U. Khan, L. Wang, N. Min-Allah and S.A. Madani *et al*., 2011. An application of Markov jump process model for activity-based indoor mobility prediction in wireless networks. Proceedings of the Frontiers Information Technology, Dec. 19-21, IEEE Xplore Press, Islamabad, Pakistan, pp: 51-56. DOI: 10.1109/FIT.2011.17

Kuo, T.W. and A.K. Mok, 1991. Load adjustment in adaptive real-time systems. Proceedings of the IEEE Real-Time Systems Symposium, Dec. 4-6, IEEE Xplore Press, San Antonio, TX, USA, pp: 160-171. DOI: 10.1109/REAL.1991.160369

Kuo, T.W., L.P. Chang, Y.H. Liu and K.J. Lin, 2003. Efficient online schedulability tests for real-time systems. IEEE Trans. Software Eng., 29: 734-751. DOI: 10.1109/TSE.2003.1223647

Lehoczky, J.P., L. Sha and Y. Ding, 1989. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. Proceedings of the IEEE Real-Time System Symposium, Dec. 5-7, IEEE Xplore Press, Santa Monica, CA, USA, pp: 166-171. DOI: 10.1109/REAL.1989.63567

Leung, J.Y.T. and J. Whitehead, 1982. On the complexity of fixed-priority scheduling of periodic, real-time tasks. Perf. Eval., 2: 237-250. DOI: 10.1016/0166-5316(82)90024-4

Liu, C.L. and J.W. Layland, 1973. Scheduling algorithms for multiprogramming in a hard real-time environment. J. ACM, 20: 40-61. DOI: 10.1145/321738.321743

Lyu, Y., L. Chen, C. Zhang, D. Qu and N. Min-Allah *et al*., 2018. An interleaved depth-first search method for the linear optimization problem with disjunctive constraints. J. Global Optimiz., 70: 737-756. DOI: 10.1007/s10898-017-0602-1

Min-Allah, N. and S.U. Khan, 2011. A hybrid test for faster feasibility analysis of periodic tasks. Int. J. Innovative Comput. Inform. Control.

Min-Allah, N., 2019. Effect of ordered set on feasibility analysis of static priority system. J. Supercomput. DOI: 10.1007/s11227-018-02742-0

Min-Allah, N., I. Ali, J. Xing and Y. Wang, 2010. Utilization bound for periodic task set with composite deadline. Comput. Electrical Eng., 36: 1101-1109. DOI: 10.1016/j.compeleceng.2010.04.003

Min-Allah, N., S.U. Khan, N. Ghani, J. Li and L. Wang *et al*., 2012. A comparative study of rate monotonic schedulability tests. J. Supercomput., 59: 1419-1430. DOI: 10.1007/s11227-011-0554-z

Min-Allah, N., S.U. Khan, X. Wang and A.Y. Zomaya, 2013. Lowest priority first based feasibility analysis of real-time systems. J. Parallel Distributed Comput., 73: 1066-1075. DOI: 10.1016/j.jpdc.2013.03.016

Sjodin, M. and H. Hansson, 1998. Improved response-time analysis calculations. Proceedings of the 19th IEEE Real-Time Systems Symposium, Dec. 4-4, IEEE Xplore Press, Madrid, Spain, pp: 399-409. DOI: 10.1109/REAL.1998.739773