RTES
Final Project Report

Schedulability Analysis for Real-Time Systems with EDF Scheduling(Fengxiang Zhang, Student Member, IEEE, and Alan Burns, Senior Member, IEEE)

Rohan Gupta -2020112022
Keerthi Pothalaraju – 2020102010

Github link - https://github.com/guptarohan6502/RTES_project

Earliest Deadline First (EDF) is an optimal scheduling algorithm for uniprocessor real-time systems. It was introduced by Liu and Layland in 1973.
On a uniprocessor, if a real-time task set cannot be scheduled by EDF, then this task set cannot be scheduled by any algorithm.

Problem Statement:

Existing results on an exact schedulability test for EDF task systems with arbitrary relative deadlines need to calculate the processor demand of the task set at every absolute deadline to check if there is an overflow in a specified time interval.
The resulting large number of calculations severely restricts the use of EDF in practice.

Aim of the paper:

This paper proposes the Quick convergence Processor-Demand Analysis (QPA) algorithm which is a new approach to check the schedulabilty analysis of an EDF system. The new results on necessary and sufficient schedulability analysis for EDF scheduling reduce the calculation times exponentially for schedulable task sets, and in most situations, for unschedulable task sets also.

Earlier Methods:

If relative deadline is less than equal to preriod, then the utilization of the task must be less than or equal to 1. Taskset is schedulable if and only if U <= 1.

If Δ is the density of the taskset. The system is said to be feasible if Δ ≤ 1. The system may not be feasible if Δ > 1.

A task set is schedulable if and only if ∀t ∈ P, h(t) ≤ t, where P = {dk|dk = kTi + Di ^ dk < min (La Lb), k ∈ N}

In the above equation, h(t) is the processor demand function. It calculates the maximum execution time requirement of all tasks' jobs which have both their arrival times and their deadlines in a contiguous interval of length t.

$$h(t) = \sum_{i=1}^{n} \max\{0, 1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\} C_i \ .$$

The bound La is defined as

$$L_a = \max\left\{D_1, \ldots, D_n, \max_{1 \le i \le n}\{T_i - D_i\}\frac{U}{1-U}\right\}.$$

The Lb bound is calculated based on the principle that a task set is feasible iff EDF algorithm can successfully schedule it during the synchronous busy period.

$$w^0 = \sum_{i=1}^{n} C_i, \quad \text{and} \quad w^{m+1} = \sum_{i=1}^{n} \left\lceil \frac{w^m}{T_i} \right\rceil C_i, \quad ,$$

The recurrence stops when w0 = w(m+1) and Lb = w(m + 1)

QPA Algortihm:

The paper proposes the QPA (Quick convergence processor demand analysis. It is a necessary and sufficient method to check the schedulability analysis with EDF scheduling. By the proposed algorithm, we do not check every deadline, and we do not need to compute all the values of deadlines in the interval even when the task set is schedulable.

- ⇨ QPA algorithm

Let L = min(La, Lb)

QPA works by starting with a value of t close to L, and then, iterating back through a simple expression towards 0.

The initial value of t is set to the maximum absolute deadline just before L.

The value of this t sequence converges for an unschedulable system to dΔ, and converges for a schedulable system to 0.

QPA stops when h(t) > t or h(t) <= $d_{\min}$

The pseudo algorithm for the QPA algorithm is as follows

```
$t \leftarrow \max\{d_i | d_i < L\}$;
while ($h(t) \le t \wedge h(t) > d_{\min}$)
  {if ($h(t) < t$) $t \leftarrow h(t)$;
   else $t \leftarrow \max\{d_i | d_i < t\}$;
  }
if ($h(t) \le d_{\min}$) the task set is schedulable;
else the task set is not schedulable;
```

The process repeats until h(t) ≤ dmin. A taskset is schedulable iff h(t) ≤ t. Therefore the value of t keeps iterating backwards until it finds a value of t that does not satisfy the condition. If h(t) = t , then the algorithm will go into an infinte loop. Hence, when t = h(t), we assign the new value of t to be the maximum absolute deadline just smaller than t.

Experiments and Evaluations:

System Model

1. We compare the number of calculations required by the original approach with upper bounds L_a; L_b, and L_a^+, and the QPA algorithm.
2. Task generation policies can significantly affect experimental results, we give the details of the policies we used in the experiments as follow

- **Utilizations generation policy**: **UUniFast algorithm**
- **Periods generation policy**:We use the approach recommended by Davis and Burns [17] to generate the task periods according to an exponential distribution.
- **Relative Deadline generation policy** The relative deadline of each task Di is generated randomly from[a,b], where a is the lower bound value of D_i, and b is the upper bound value of D_i, such that if

$\mathbf{C_i < 10}$; a = Ci;
when $\mathbf{10 < C_i < 100}$; a = 2 *C_i;
when **100<Ci < 1000**; a = 3*C_i; and
when $\mathbf{C_i < 1000}$; a= 4*Ci.
The default value $\mathbf{b = 1.2 * T_i}$.

Measurement Metric

- A reasonable metric to compare results is to measure the number of times the processor demand function h(t)(processor demand function) has to be calculated
- We do the experiments separately for schedulable and unschedulable task
- When we experiment on schedulable task sets, if a generated task set is unschedulable, the program discards it and does not count it into the experimental results. However, all experiments use the same default generation policy described above (unless an alternative is specified)
- H(t) processor demand function is calculated according to the QPA algorithm and La, Lb, La_star method.

Experiments on Schedulable Taskset:

Taskset – Set of n tasks
Tasks – task is a unit of execution or a unit of work

Random tasksets are generated according to the task generation policy. Density of the taskset is calculated and if density (Δ) >1 the taskset is discarded and new random taskset is generated.

The output of each experiment is the average of randomly generated 60 schedulable taskset.

1. Impact of Number of Tasks

Utilization = 0.9
T_max/T_min = 1000(T_i = Period of Task i)

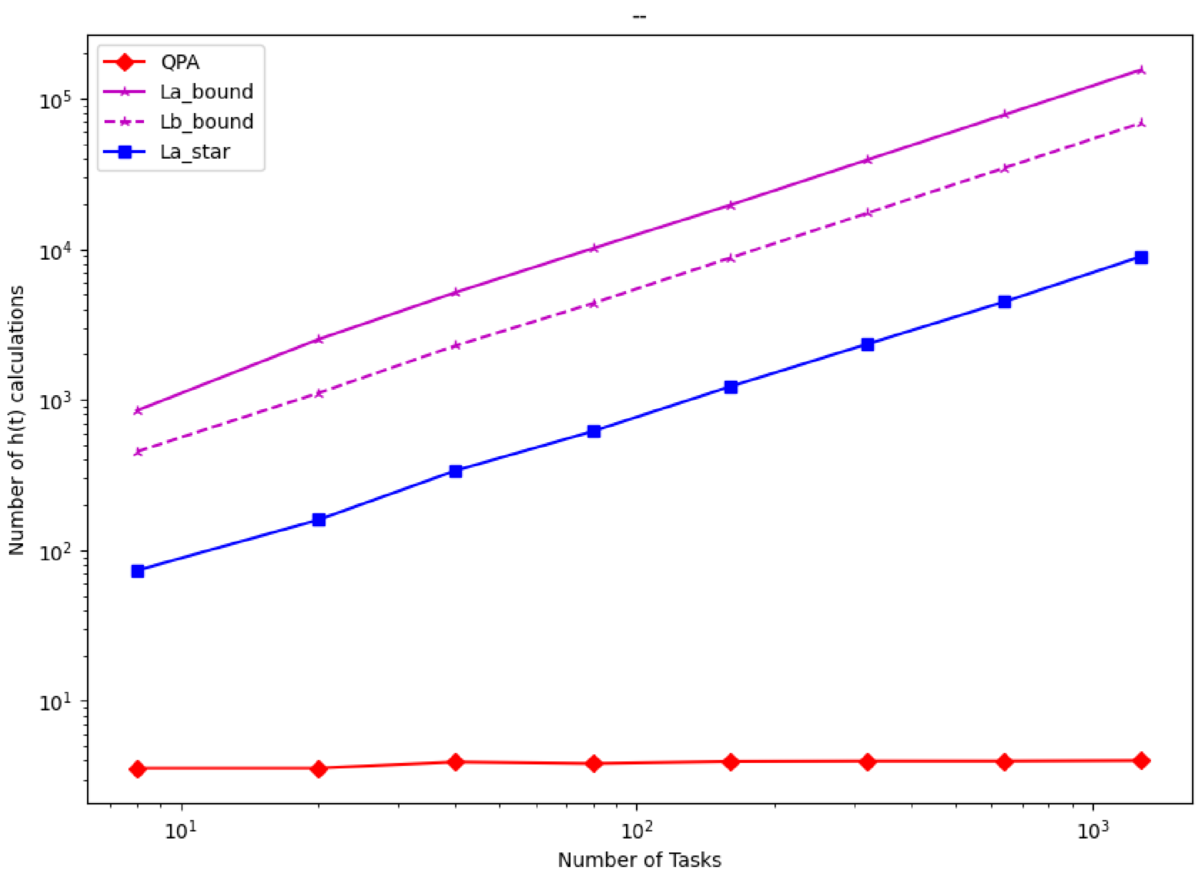
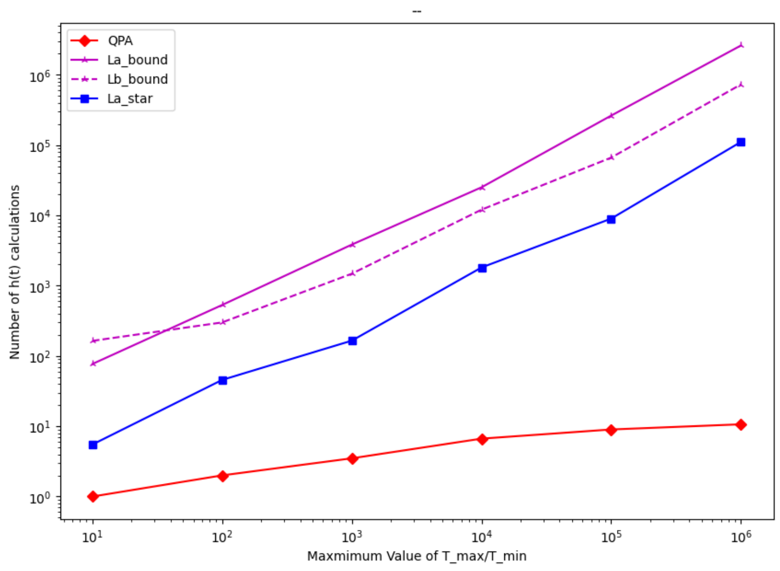


Fig: Impact of Number of Tasks

2. <u>Impact of the Task Periods Range</u>

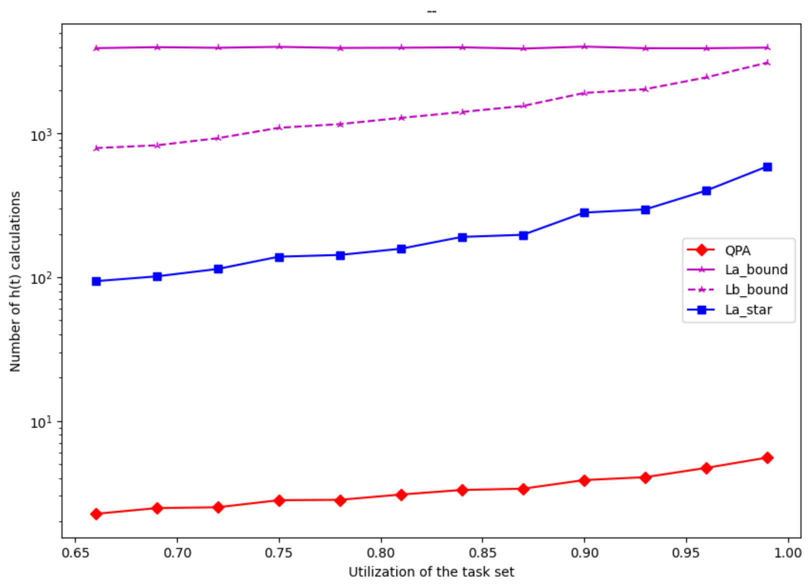
Number of tasks for each task set is 30
utilization = 0.9.
We Vary T_max/T_min and calculate number of h(t) caclualtions required.



<u>Fig: Impact of the Task Periods Range</u>

3. <u>Impact of the Utilizations</u>

Number of tasks for each task set is 30
T_max/T_min = 1000
We Vary utilizations from 0 – 0.99 and calculate number of h(t) caclualtions required.



<u>Fig: Impact of the Utilizations</u>

Experiments on Unschedulable Taskset:

Taskset – Set of n tasks
Tasks – task is a unit of execution or a unit of work

Random tasksets are generated according to the task generation policy. Density of the taskset is calculated and if density (Δ) <=1 the taskset is discarded and new random taskset is generated.

The output of each experiment is the average of randomly generated 60 **unschedulable** taskset.

4. Impact of Number of Tasks(Unschedulable Taskset)

Utilization = 0.9
T_max/T_min = 1000(T_i = Period of Task i)

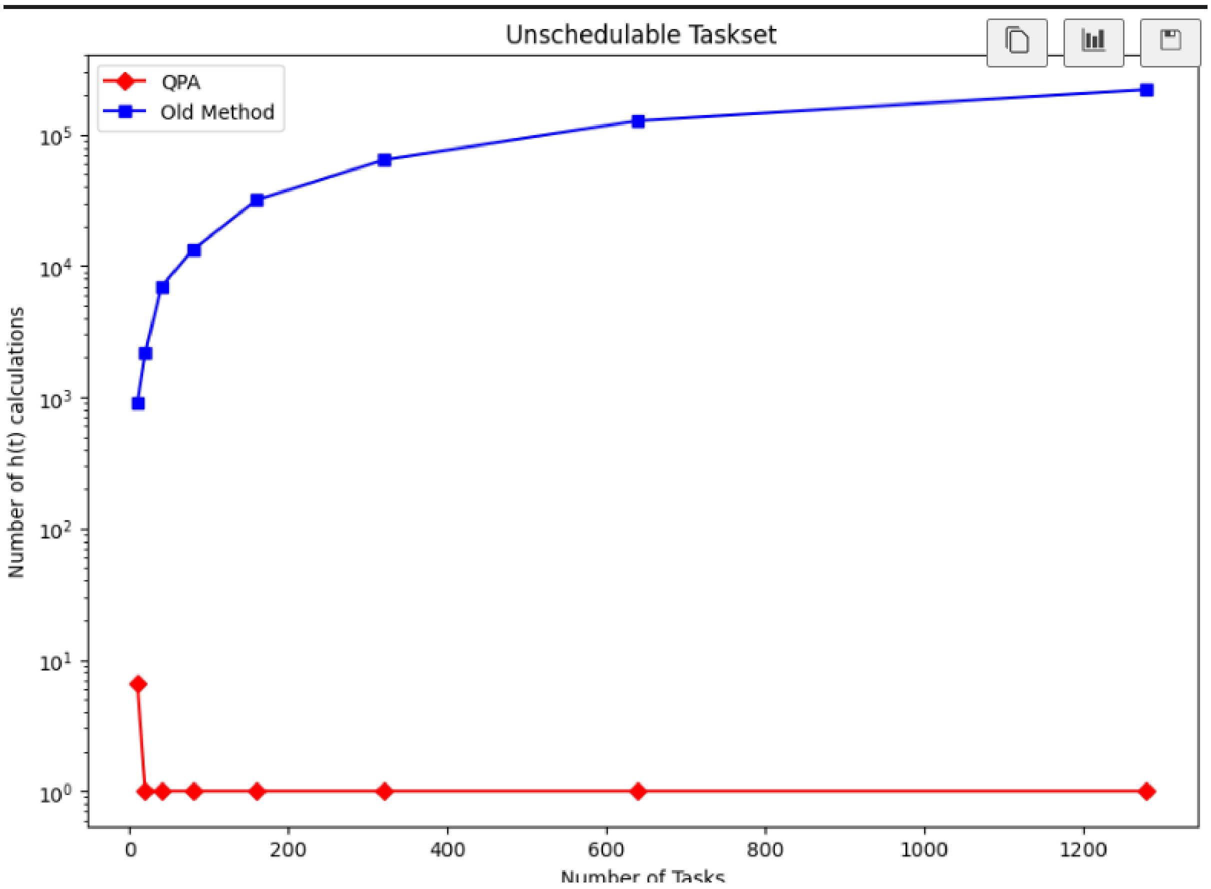
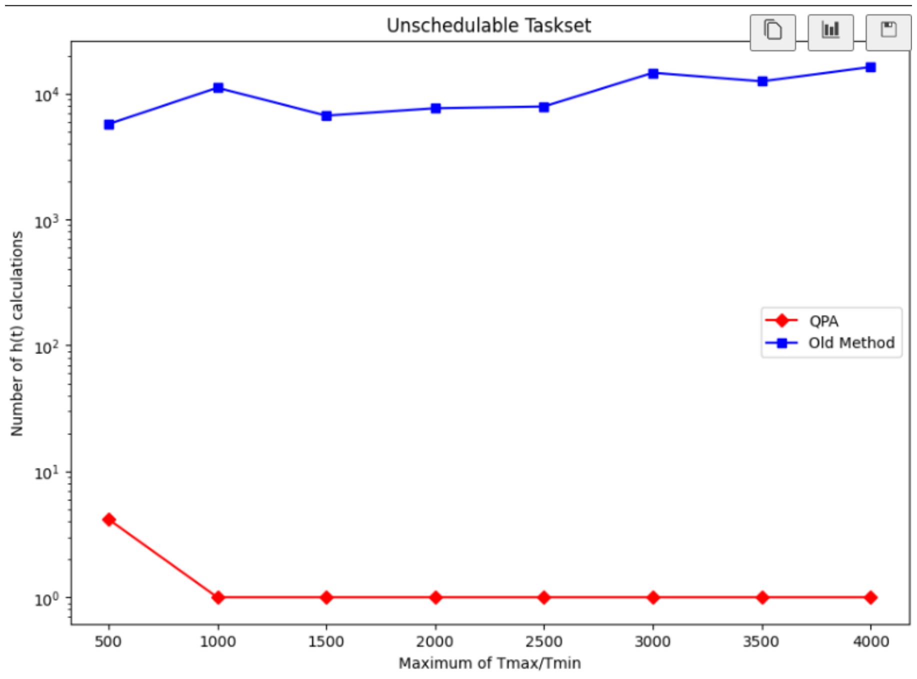


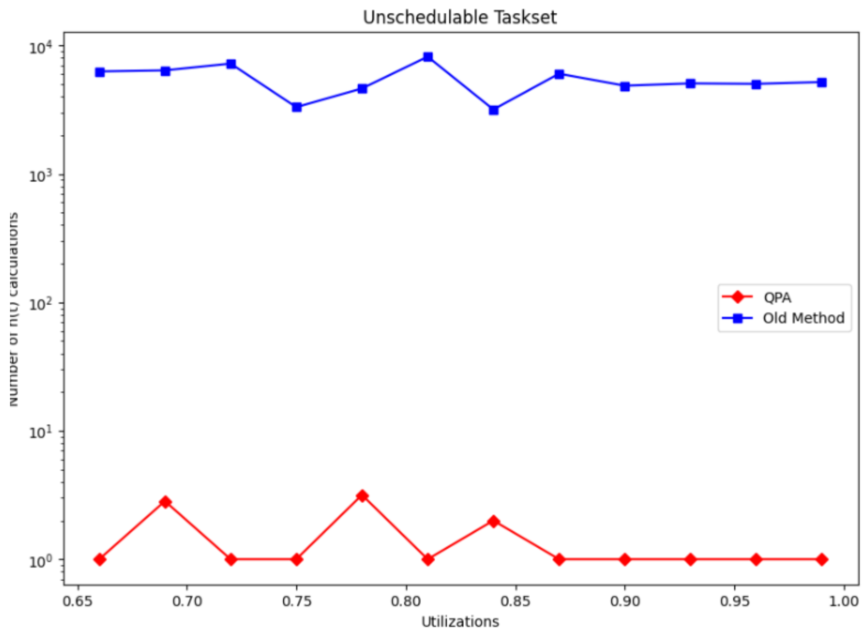
Fig: Impact of Number of Tasks

5. **<u>Impact of the Task Periods Range(Unschedulable)</u>**
   Number of tasks for each task set is 30
   utilization = 0.9.
   We Vary T_max/T_min and calculate number of h(t) calculations required.



<u>Fig: Impact of the Task Periods Range</u>

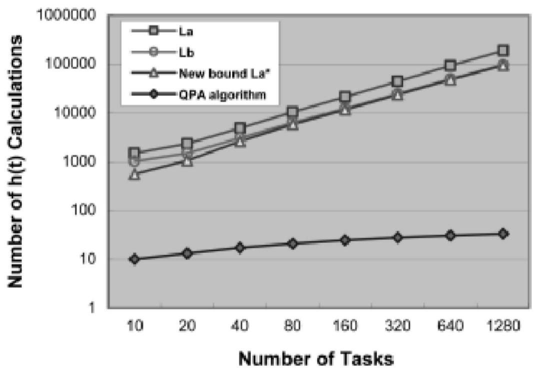
1. **<u>Impact of the Utilizations</u>**
   Number of tasks for each task set is 30
   T_max/T_min = 1000
   We Vary utilizations from 0 – 0.99 and calculate number of h(t) caclualtions required.



<u>Fig: Impact of the Utilizations</u>

Results Given in Paper:

1. **Schedulable Taskset**



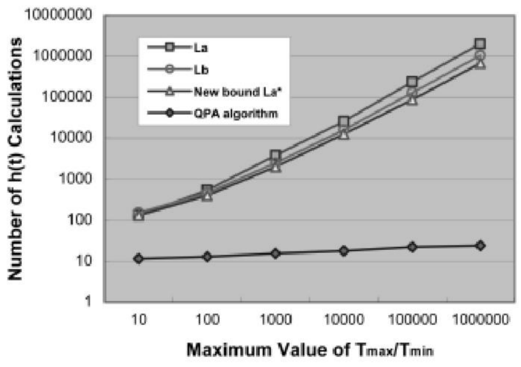
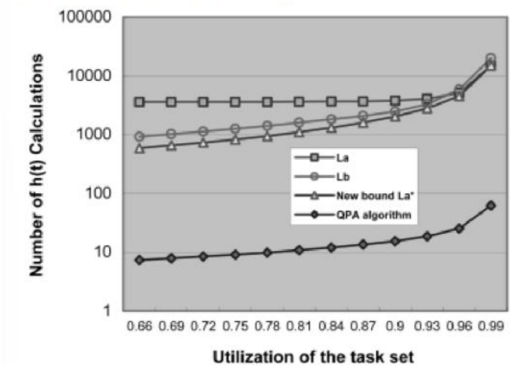
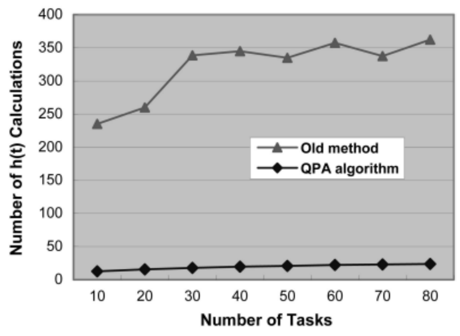
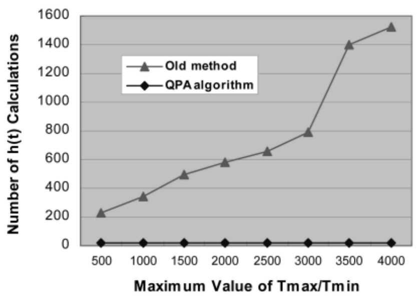


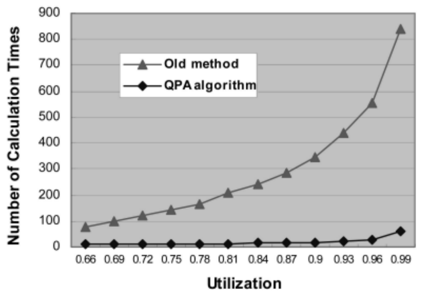
Fig. 3. Impact of the task periods range.



2. **Unschedulable Taskset**







Our Results match with the results given in paper for schedulabe taskset and are somewhat similar in the case of unschedulable taskset.