Schedulability Analysis for Real-Time Systems with EDF Scheduling

Fengxiang Zhang, Student Member, IEEE, and Alan Burns, Senior Member, IEEE

Keerthi Pothalaraju – 2020102010

Rohan Gupta - 2020112022

Github link

Earliest Deadline First(EDF) Algortihm

Earliest Deadline First (EDF) is an optimal scheduling algorithm for uniprocessor real-time systems. It was introduced by Liu and Layland in 1973.

According to the EDF algorithm, an arrived job with the earliest absolute deadline is executed first. It is the most common dynamic scheduling algorithm for real-time systems.

EDF guarantees that all deadlines are met provided that total CPU utilization(U) < 1.

on a uniprocessor, if a real-time task set cannot be scheduled by EDF, then this task set cannot be scheduled by any algorithm.

Why are schedulability test for real time systems necessary?

- In online systems during the run-time of a system there could be new tasks arrive that need (if possible) to be added to the task set. **The system must recalculate schedulability online to decide whether to allow the new tasks to enter into the system.** Such online admission control gives a much higher requirement for the performance of the schedulability test as the decisions have to be made in a very short time and should not occupy too much system resource

Previous Results on Exact Schedulability Analysis

- In online systems during the run-time of a system there could be new tasks arrive that need (if possible) to be added to the task set. **The system must recalculate schedulability online to decide whether to allow the new tasks to enter into the system.** Such online admission control gives a much higher requirement for the performance of the schedulability test as the decisions have to be made in a very short time and should not occupy too much system resource

Evolution of schedulability tests

1. Liu and Layland presented a necessary and sufficient schedulability condition

- **Assumption**: Relative deadlines are equal to their periods.
- **Schedulability condition**: Utilization of the task must be less than or equal to
  1. Taskset is schedulable if and only if U <= 1

2. Approximate or sufficient test for EDF scheduling :

Assumption: Relative deadlines are less than their periods.

The density of a taskset is defined as delta(Δ) = $\sum_{k=1}^{n} \frac{e_k}{\min(D_k, p_k)}$

- The system is said to be feasible if Δ <= 1
- The system may not be feasible for Δ > 1

Problem statement

- The schedulablility test decides if a given task set can be scheduled, such that no tasks in the taskset miss their deadlines.
- On a uniprocessor, if a taskset cannot be scheduled by EDF, then it cannot be scheduled by any other algorithm. Hence, we need reliable schedulability tests for EDF.
- 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.

- QPA provides fast schedulability test for arbitrary relative deadline EDF systems.
- QPA builds on the traditional processor demand analysis.

Exact schedulability analysis for EDF systems

A set of **periodic** tasks is schedulable if and only if all absolute deadlines in the interval

[0, max{s_i} + 2H] are met, where s_i is the start time of task t_i, and H is hyperperiod.

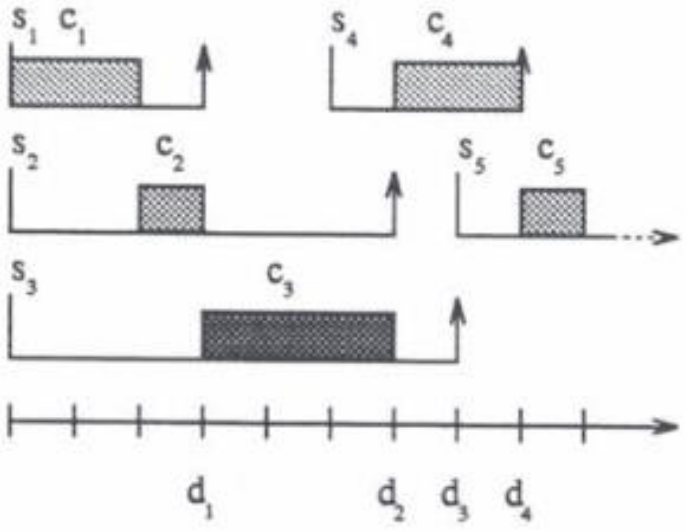
Similarily, extending this to the sporadic task set, a sporadic taskset is schedulable if and only if h(t) <= t, where h(t) is the processor demand function.

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 amount of time required until t, to execute all the tasks have their arriva; times and deadlines within t interval

Visualization of h(t)



Activation arrangement.

$$
\begin{aligned}
c_1 &= H_\tau(d_1) \\
&\vdots \quad \vdots \quad \vdots \\
c_x + \cdots + c_2 + c_1 &= H_\tau(d_x) \\
&\vdots \quad \vdots \quad \vdots
\end{aligned}
$$

- h(t) represents the amount of computation time that has been requested by all the jobs whose deadline is less or equal to t.

Proof (Preemptively scheduling hard-real-time sporadic tasks on one processor)

Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier

- Task is **not feasible** iff there exists an integer t >= 0 such that h(t) > t
- If $\sum_{i=1}^{n} e_i/p_i > 1$. We will show that there exists a t such that h(t) > t.
- Choose $\{t \geq \max_{1 \leq i \leq n}\{d_i\}$

$h_{\mathcal{R}}(t)$

$= \{\text{By definition}\}$

$$\sum_{i=1}^{n} e_i \cdot \max\{0, \lfloor \frac{t-d_i}{p_i} \rfloor + 1\}$$

$= \{t \geq \max_{1 \leq i \leq n}\{d_i\} \Rightarrow \lfloor \frac{t-d_i}{p_i} \rfloor \geq 0\}$

$$\sum_{i=1}^{n} e_i \cdot (\lfloor \frac{t-d_i}{p_i} \rfloor + 1)$$

$= \{$ Pulling the 1 inside the floor$\}$

$$\sum_{i=1}^{n} e_i \cdot \lfloor \frac{t+p_i-d_i}{p_i} \rfloor$$

$= \{$Since $p_i|t$, we pull t/p_i from within the floor function, and rearrange terms$\}$

$$t \cdot \sum_{i=1}^{n} \frac{e_i}{p_i} + \sum_{i=1}^{n} e_i \cdot \lfloor \frac{p_i-d_i}{p_i} \rfloor$$

$= \{$rearranging terms$\}$

$$t \cdot (1 + \sum_{i=1}^{n} \frac{e_i}{p_i} - 1) + \sum_{i=1}^{n} e_i \cdot \lfloor \frac{p_i-d_i}{p_i} \rfloor$$

$= \{$rearranging terms$\}$

$$t + t \cdot (\sum_{i=1}^{n} \frac{e_i}{p_i} - 1) + \sum_{i=1}^{n} e_i \cdot \lfloor \frac{p_i-d_i}{p_i} \rfloor$$

$> \{\text{Substituting } -\sum_{i=1}^{n} e_i \cdot \lfloor \frac{p_i - d_i}{p_i} \rfloor / (\sum_{i=1}^{n} \frac{e_i}{p_i} - 1) \text{ for the latter } t\}$

$$t + \frac{(-\sum_{1=1}^{n} e_i \lfloor \frac{p_i - d_i}{p_i} \rfloor)}{(\sum_{1=1}^{n} \frac{e_i}{p_i} - 1)} \cdot$$

$$(\sum_{1=1}^{n} \frac{e_i}{p_i} - 1) + \sum_{i=1}^{n} e_i \cdot \lfloor \frac{p_i - d_i}{p_i} \rfloor$$

$= \{\text{cancelling the common factor } (\sum_{i=1}^{n} \frac{e_i}{p_i} - 1)\}$

$$t - \sum_{i=1}^{n} e_i \cdot \lfloor \frac{p_i - d_i}{p_i} \rfloor + \sum_{i=1}^{n} e_i \cdot \lfloor \frac{p_i - d_i}{p_i} \rfloor$$

$= \{\text{Cancelling terms}\}$

$$t.$$

Hence, $h_{\mathcal{R}}(t) > t$.

From the proof we can see that a taskset is schedulable if, h(t) ≤ t.

Theorem : La bound

- A general task set is schedulable if and only if U ≤ 1 and **∀ t** < L_a, t < h(t)

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

This bound is for a general taskset. There need not be any relation between Ti and Di.

Earlier we had to calculate for all the absolute deadlines until time t to check if the taskset was schedulable or not. With the La bound, It is enough to check for all the absolute deadlines until La. This reduces the number of computations to check the schedulability analysis of the system.

Proof

Lemma 6 *Suppose τ is not feasible and $\sum_{i=1}^{n} \frac{e_i}{p_i} < 1$. Then $h_{\mathcal{R}}(t) > t$ implies $t < \max_{1 \le i \le n}\{d_i\}$ or $t < \max_{1 \le i \le n}\{p_i - d_i\} \cdot \sum_{i=1}^{n} \frac{e_i}{p_i} / (1 - \sum_{i=1}^{n} \frac{e_i}{p_i})$.*

Proof : Assume that τ is not feasible, $\sum_{i=1}^{n} \frac{e_i}{p_i} < 1$, $h_{\mathcal{R}}(t) > t$, and $t \ge \max_{1 \le i \le n}\{d_i\}$. Then our proof obligation is to show that t must be less than $\max_{1 \le i \le n}\{p_i - d_i\} \cdot \sum_{i=1}^{n} \frac{e_i}{p_i} / (1 - \sum_{i=1}^{n} \frac{e_i}{p_i})$.

$h_{\mathcal{R}}(t)$

$= \{$By definition of $h_{\mathcal{R}}(t)\}$

$$\sum_{i=1}^{n} e_i \cdot \max\{0, \lfloor \frac{t - d_i}{p_i} \rfloor + 1\}$$

$\le \{$Eliminating the floor function, and rearranging terms$\}$

$$\sum_{i=1}^{n} e_i \cdot (\frac{t + p_i - d_i}{p_i})$$

$= \{$Rearranging terms $\}$

$$t \cdot \sum_{i=1}^{n} \frac{e_i}{p_i} + \sum_{i=1}^{n} \frac{e_i}{p_i}(p_i - d_i)$$

$\le \{$since $\max_{1 \le i \le n}\{p_i - d_i\} \ge (p_j - d_j)$ for all j, $1 \le j \le n\}$

$$t \cdot \sum_{i=1}^{n} \frac{e_i}{p_i} + \max_{1 \le i \le n}\{p_i - d_i\} \sum_{i=1}^{n} \frac{e_i}{p_i}$$

Substituting t for $h_{\mathcal{R}}(t)$, and from the assumption that $h_{\mathcal{R}}(t) > t$, we obtain:

$$t < t \cdot \sum_{i=1}^{n} \frac{e_i}{p_i} + \max_{1 \le i \le n}\{p_i - d_i\} \cdot \sum_{i=1}^{n} \frac{e_i}{p_i},$$

$\equiv \{$ Rearranging terms, and factoring out $t\}$

$$t(1 - \sum_{i=1}^{n} \frac{e_i}{p_i}) < \max_{1 \le i \le n}\{p_i - d_i\} \cdot \sum_{i=1}^{n} \frac{e_i}{p_i}$$

$\equiv \{$dividing both sides by the positive quantity $(1 - \sum_{i=1}^{n} \frac{e_i}{p_i})\}$

$$t < \max_{1 \le i \le n}\{p_i - d_i\} \cdot \frac{\sum_{i=1}^{n} \frac{e_i}{p_i}}{(1 - \sum_{i=1}^{n} \frac{e_i}{p_i})}.$$

The following represents the final La bound implemented in the paper. However, this is under the assumption that Di <= Ti

Theorem : *A general task set is schedulable if and only if* $U \leq 1$ *and*

$$\forall t < L_a, h(t) \leq t,$$

where L_a *is defined as follows:*

$$L_a = \max\left\{D_1, \ldots, D_n, \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1-U}\right\}.$$

Note, $\frac{\sum_{i=1}^{n}(T_i-D_i)U_i}{1-U} \leq \max_{1\leq i\leq n}\{T_i - D_i\}\frac{U}{1-U}$.

Theorem : Lb bound

- Like La bound, this theorem gives a bound over the value of t for which we need to check the schedulability analysis for our taskset.
- a task set is feasible iff the EDF algorithm can successfully schedule it during the synchronous busy period.

Theorem 3 ([19]). *A general task set is schedulable if and only if* $U \leq 1$ *and*

$$\forall t \leq L_b, h(t) \leq t,$$

where L_b *is the length of the synchronous busy period of the task set.*

Synchronous busy period

A **synchronous busy period** is a processor busy period in which all tasks are released simultaneously at the beginning of the processor busy period, and then, at their maximum rate, and ended by the first processor idle period (the length of such a period can be zero).

Proof

To prove the above we prove that no task misses its deadline inside the intervals if no deadline is missed in the synchronous busy period.

The length of the synchronous busy period L_b can be computed by the following process

$$w^0 = \sum_{i=1}^{n} C_i,$$

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

where the recurrence stops when $w^{m+1} = w^m$, and then $L_b = w^{m+1}$.

Theorem :

- As the processor demand hðtÞ could only change at the times of absolute deadlines, the schedulability test becomes the following

Theorem 4 ([5], [7], [11], [12], [19]). *A task set is schedulable if and only if* $U \leq 1$ *and*

$$\forall t \in P, h(t) \leq t,$$

where $P = \{d_k | d_k = kT_i + D_i \wedge d_k < \min(L_a, L_b), k \in N\}$, *where* L_a *is calculated by (4) and* L_b *is the solution of (5) and (6).*

In Section 5, we will present a minor improvement to the calculation of L_a.

Proof contd..

- The length of the synchronous busy period is the maximum length of any possible busy processor
period in any schedule.
- Let all tasks be released simultaneously at t = 0, and then, at their maximum rate. According to the definition of the synchronous busy period, the processor is always busy during [0, Lb).
- Suppose h(Lb) > Lb , then the processor continues to be busy at and after t = Lb , so the busy period must be longer than L b . This contradicts point 1.
- Therefore h(Lb) <= Lb
- Therefore, for a taskset to be schedulable,
- U <= 1 and ∀ t < Lb, h(t) <= t;

- The existing results on exact schedulability analysis for EDF scheduling with arbitrary relative deadlines need tocalculate the processor demand of the task set at everyabsolute deadline to check if there is an overflow in aspecified time interval.
- This interval is bounded by a certain value which guarantees we can find a failure point if the task set is not schedulable. In such an interval, there could be a very large number of absolute deadlines that need to be verified. The significant effort required to perform the exact schedulability test restricts the use of EDF in realistic systems; hence, the EDF algorithm has not been used as widely as the fixed priority algorithms in commercial realtime systems.

QPA (Quick Convergence Processor-demand Analysis)

- QPA builds on the traditional 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.

- **Notations used :**

$$d_{\min} = \min\{D_i\}.$$

$$d^{\Delta} = \max\{d_i | 0 < d_i < L \wedge h(d_i) > d_i\}.$$

$$d^t_{\max} = \max\{d_i | d_i \leq t\}.$$

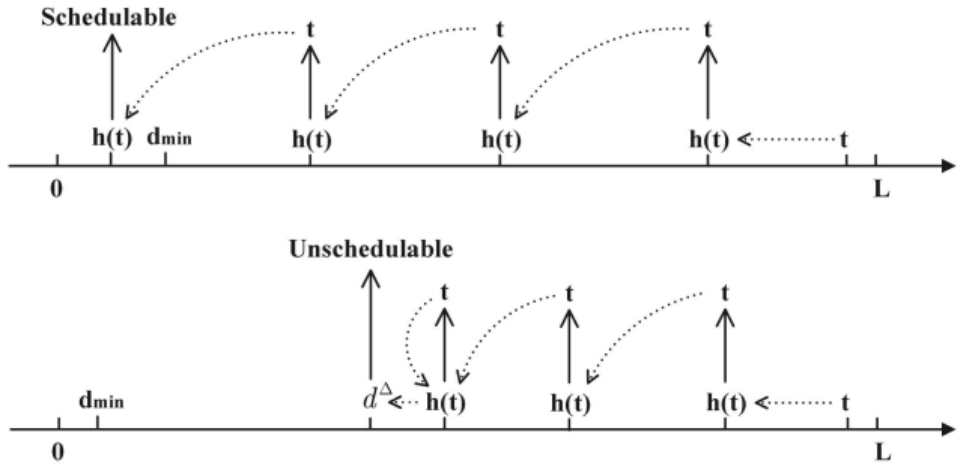
QPA Algorithm

- We define L as the minimum value of La and Lb. L is equal to Lb when U = 1

$$L = \begin{cases} \min(L_a, L_b) & U < 1 \\ L_b & U = 1 \end{cases}.$$

- QPA works by starting with a value of t close to L, and then, iterating back through a simple expression towards 0.
- The value of this t sequence converges for an unschedulable system to d, and converges for a schedulable system to 0.
- QPA stops when stopped

  once h(t) <= d_{min}



QPA Algorithm

Theorem 5. *A general task set is schedulable if and only if* $U \leq 1$ *and the result of the following iterative algorithm is* $h(t) \leq d_{\min}$.

QPA

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

Illustration of QPA

Task	Execution Time	Relative Deadline	Period
τ_1	6000	18000	31000
τ_2	2000	9000	9800
τ_3	1000	12000	17000
τ_4	90	3000	4200
τ_5	8	78	96
τ_6	2	16	12
τ_7	10	120	280
τ_8	26	160	660

Utilization :

$U \cong 0.803 \leq 1.$

La :
18000

Lb :
16984

QPA algorithm

1. $t = 16{,}974, h(t) = 8{,}890,$
2. $t = 8{,}890, h(t) = 3{,}080,$
3. $t = 3{,}080, h(t) = 1{,}098,$
4. $t = 1{,}098, h(t) = 362,$
5. $t = 362, h(t) = 118,$
6. $t = 118, h(t) = 26,$
7. $t = 26, h(t) = 2.$

Since $h(t) = 2 \leq d_{\min'}$ the task set is schedulable.

Understanding of QPA

- L = min(La, Lb)
- Therefore, the initial value of t is the maximum absolute deadline just before L. The initial value of L is 16974.

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

- H(t) = 8890
- We then take the new value of t = h(t-1)
- We repeat this process until h(t) <= dmin == 12

Theorem : L_a^* bound

Theorem 6. *A general task set is schedulable if and only if* $U \leq 1$ *and*

$$\forall t \in P, h(t) \leq t,$$

where $P = \{d_k | d_k = kT_i + D_i \wedge d_k < L_a^*, k \in N\}$, *and where*

$$L_a^* = \max\left\{(D_1 - T_1), \ldots, (D_n - T_n), \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1 - U}\right\}. \quad (9)$$

Experiments and Evaluations:

System Model

- 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.
- 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

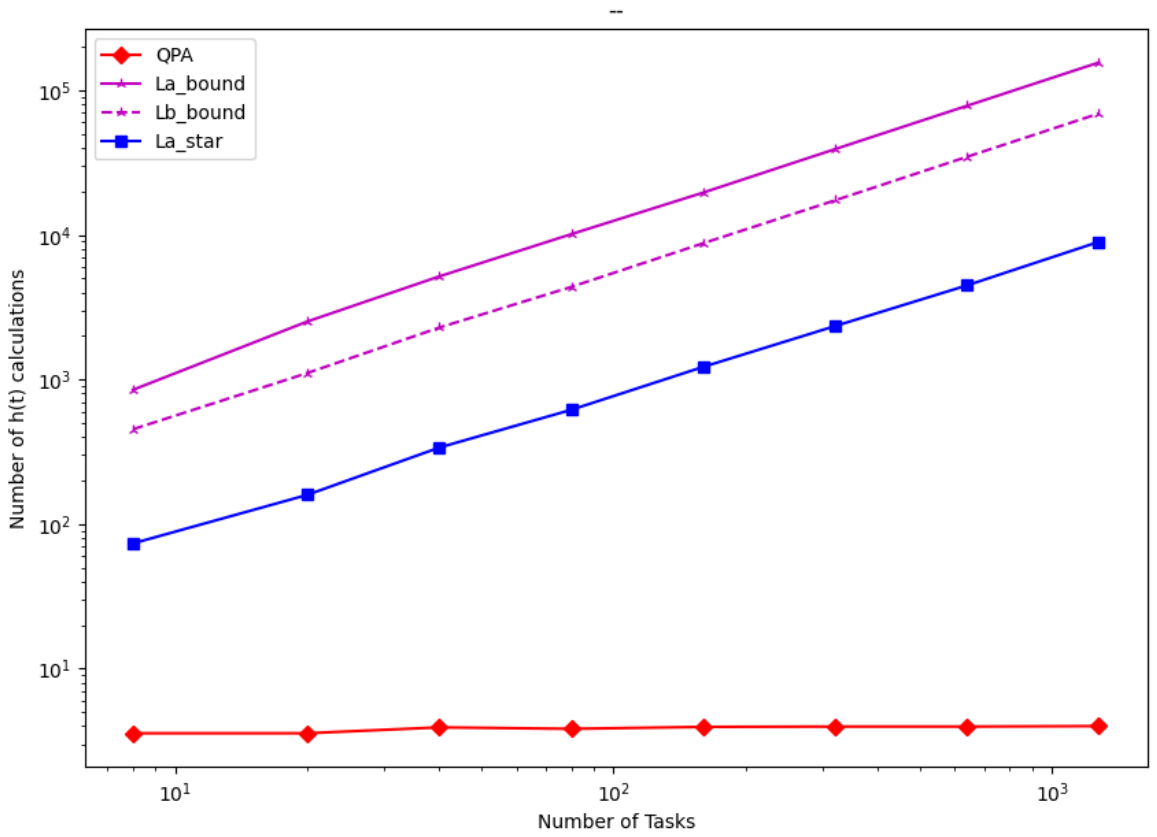
  $C_i < 10$; a = Ci;
  when **$10 < C_i < 100$**; a = 2 *C_i;
  when **$100 < Ci < 1000$**; a = 3*C_i; and
  when **$C_i < 1000$**; a= 4*Ci.
  The default value **$b = 1.2 * T_i$.**

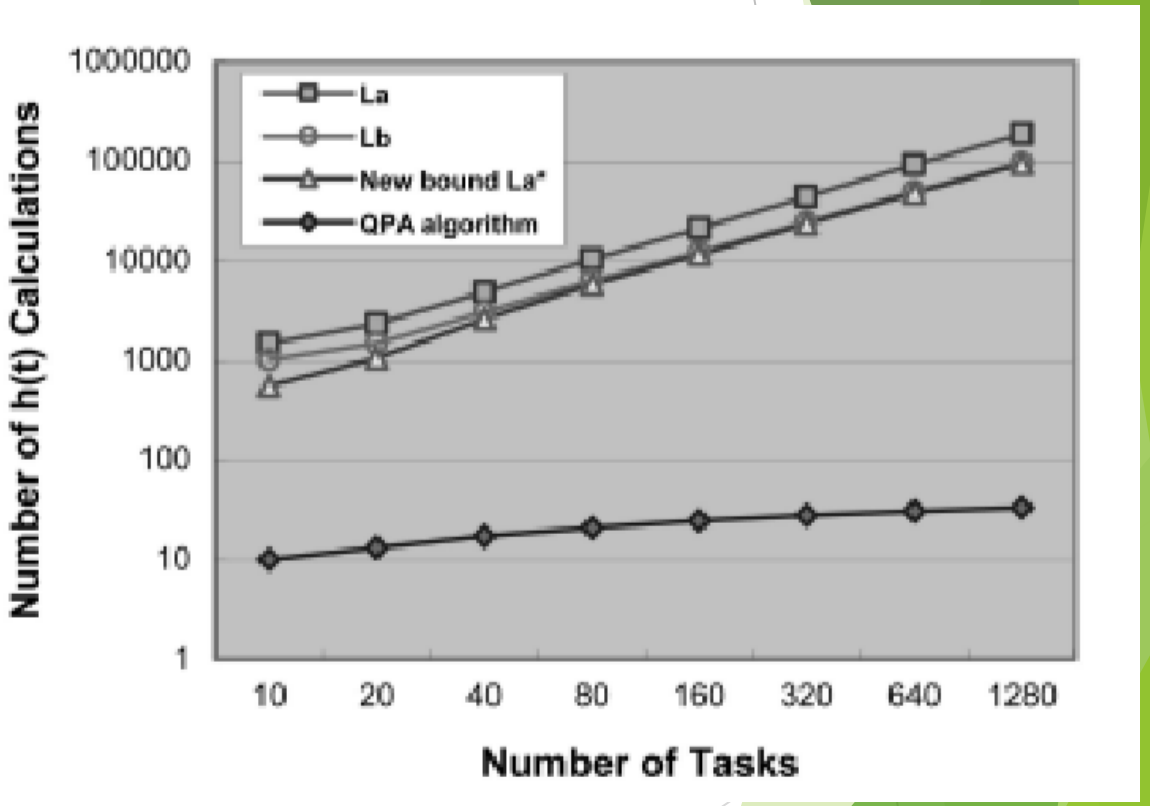
Measurement Metric

- 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)

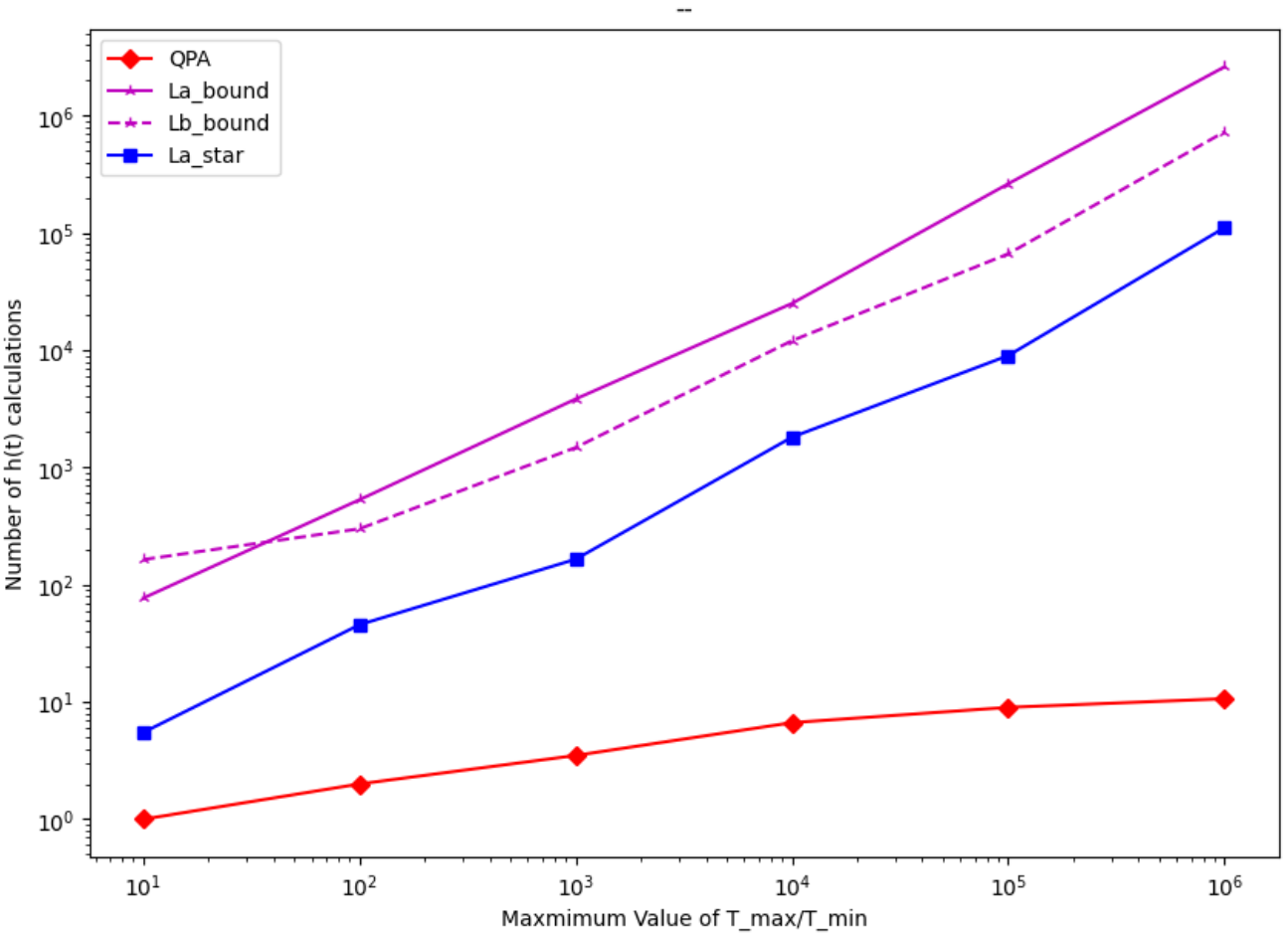
Experiments On Schedulable Task sets

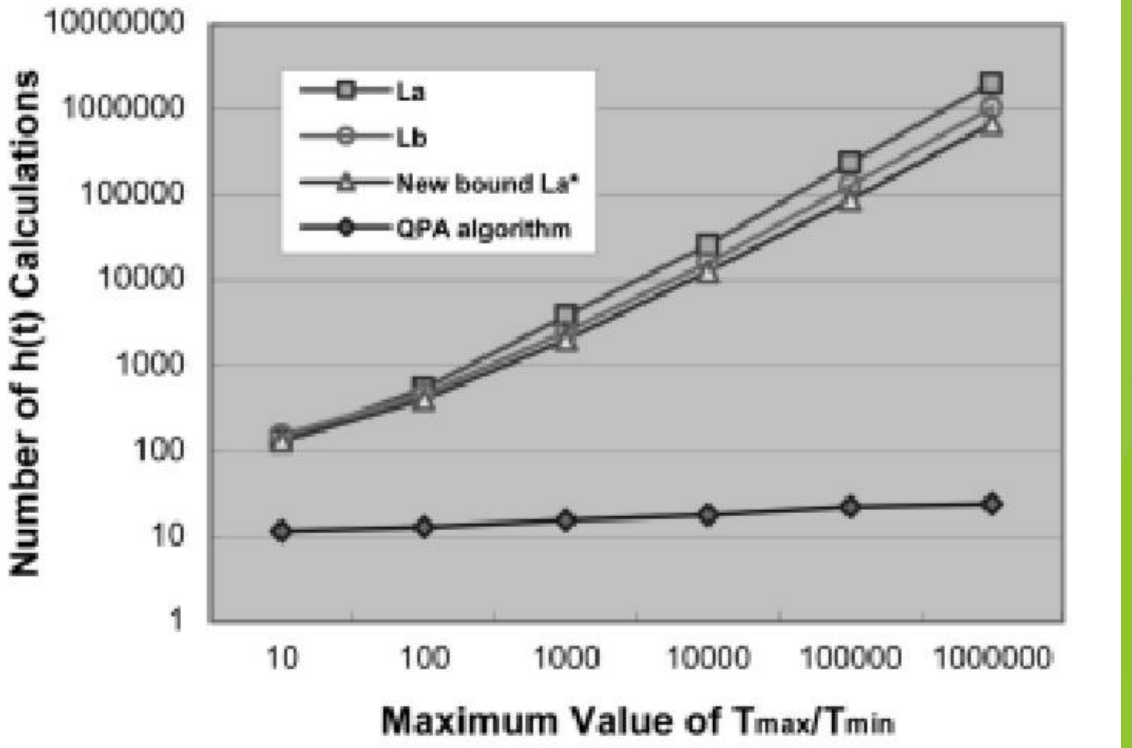
Impact of Number of Tasks



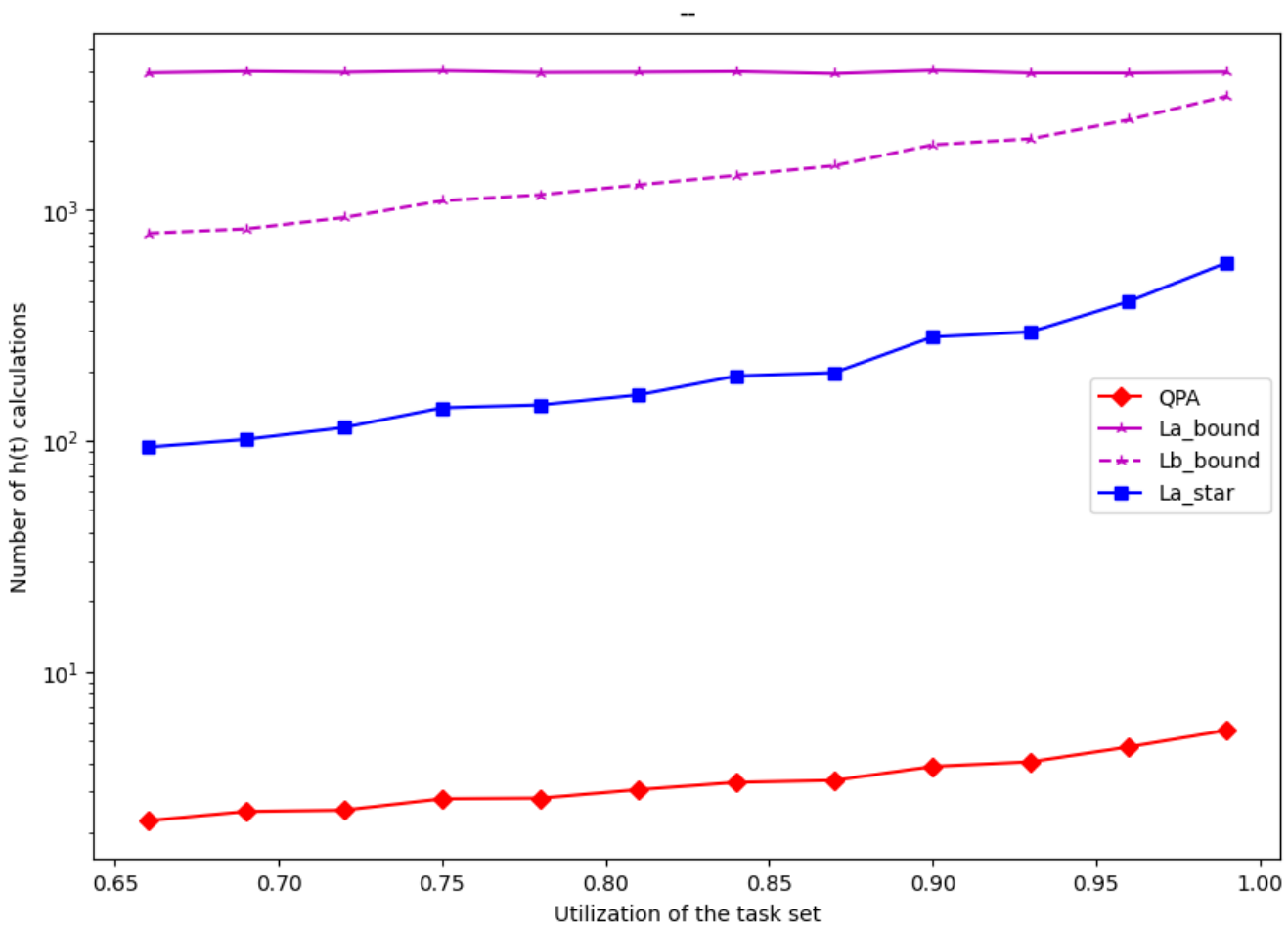


Impact of the Task Periods Range

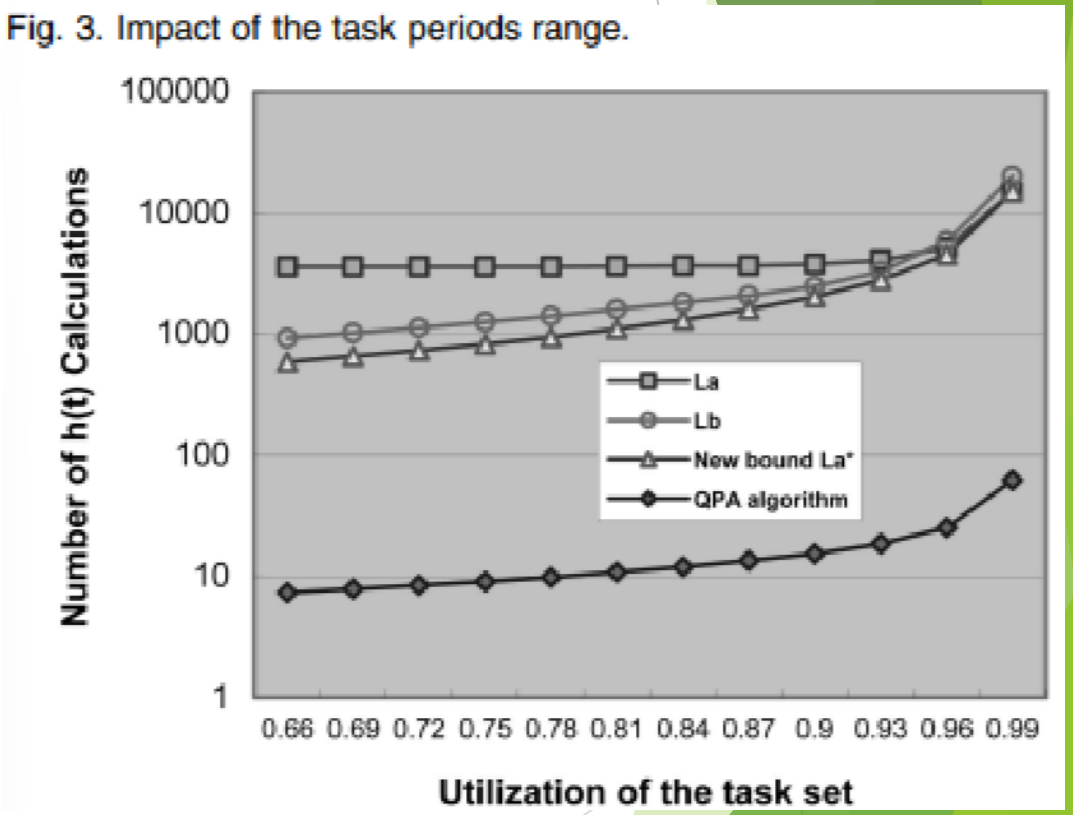




number of tasks for each task set is 30,
utilization is 0.9.

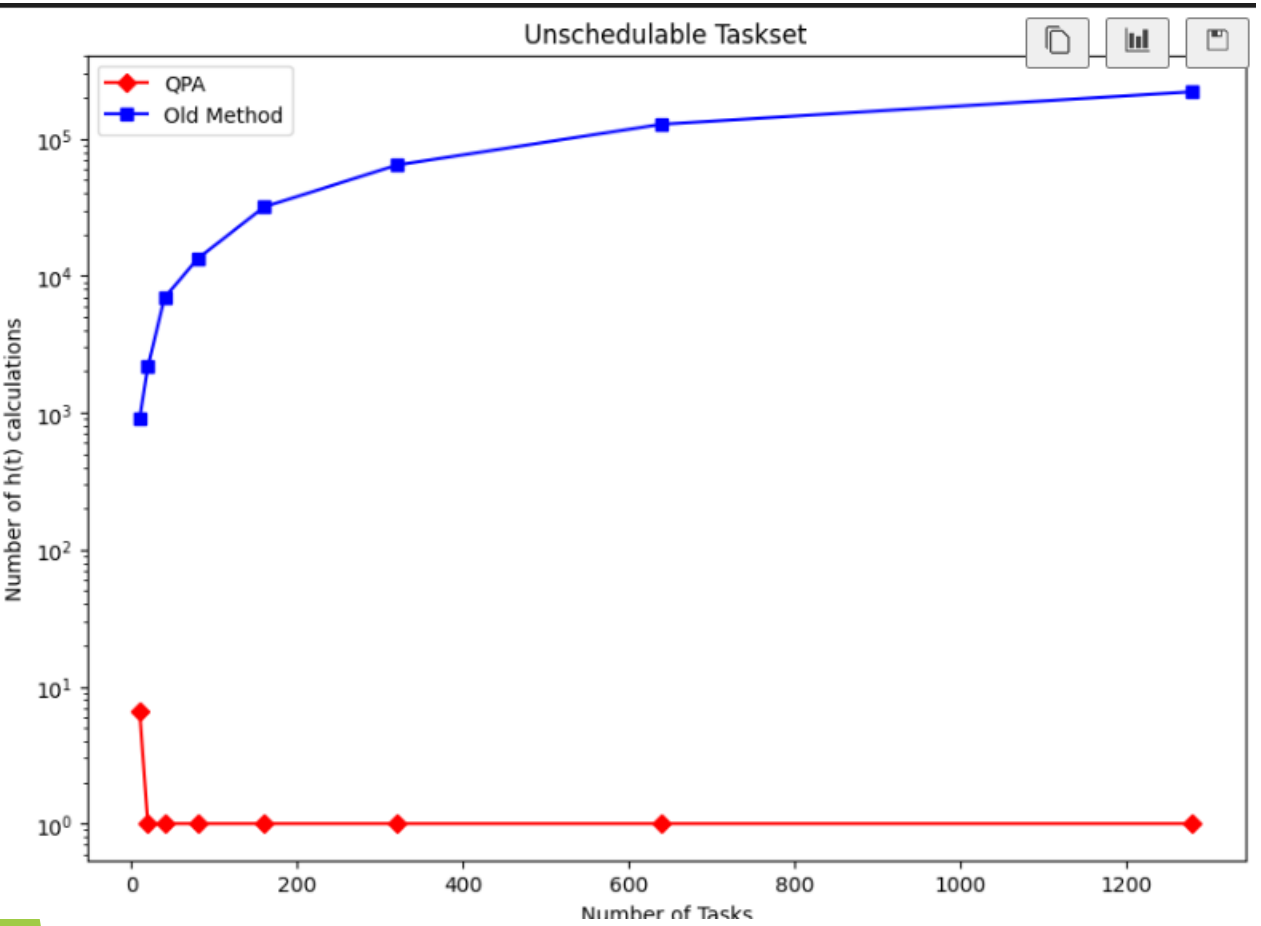


Impact of the Utilization

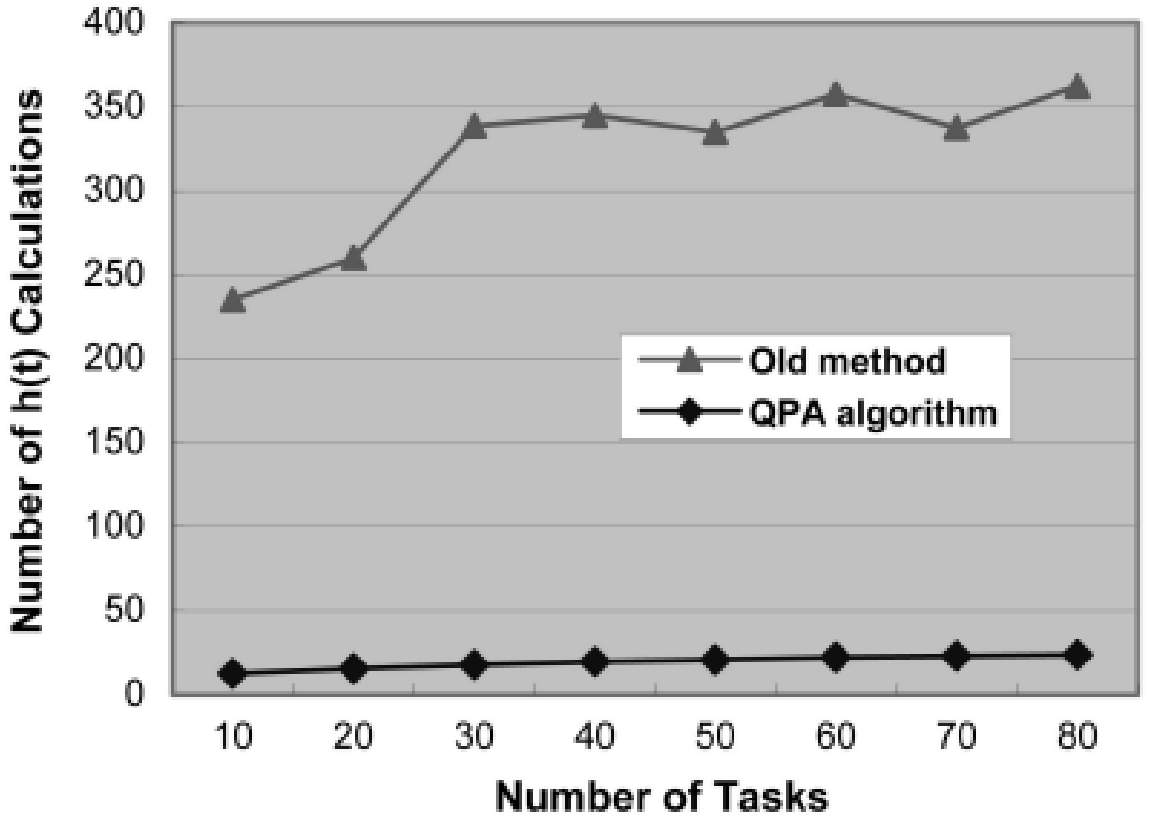


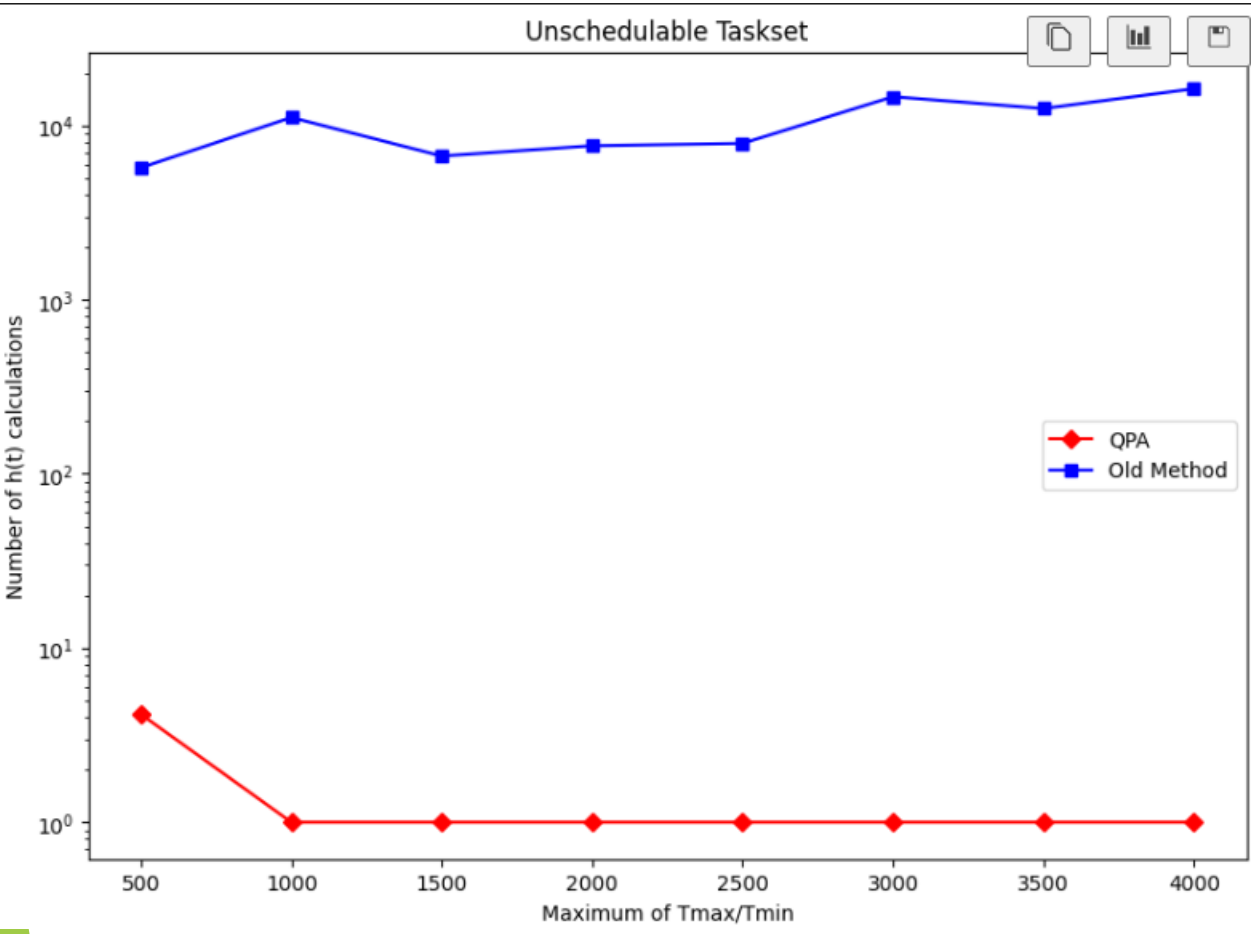
The size of each task set is 30 and
Tmax=Tmin 1;000

Experiments On UnSchedulable Task sets

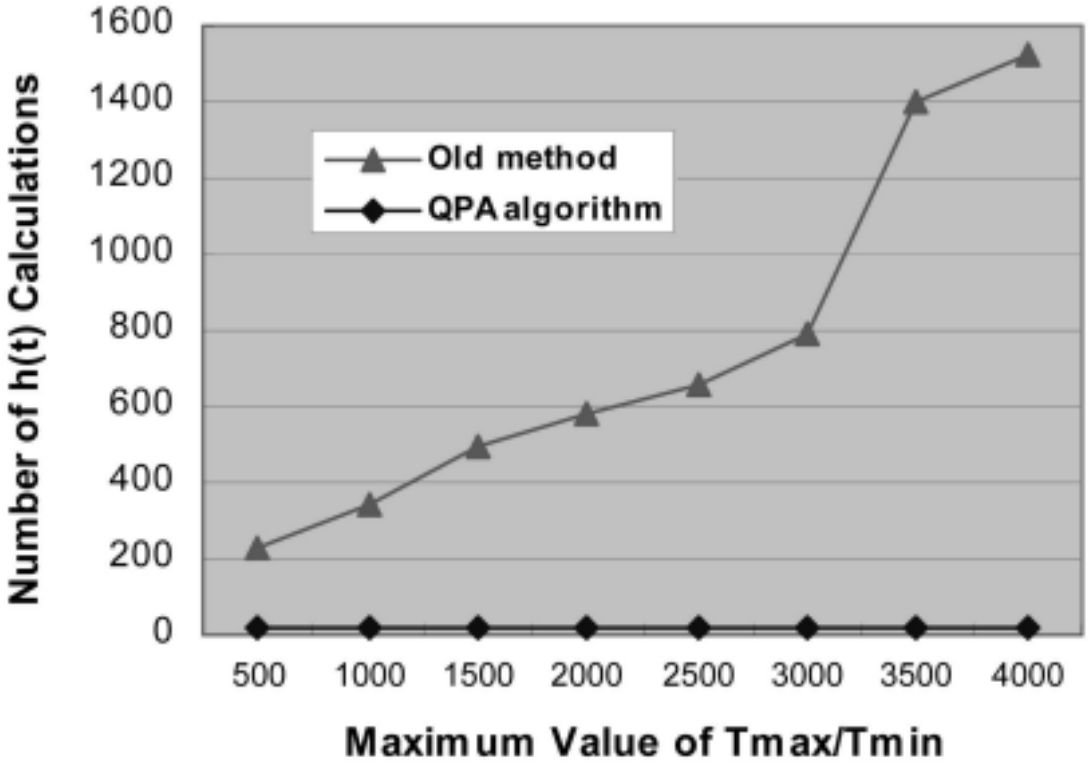


Impact of Number of Tasks

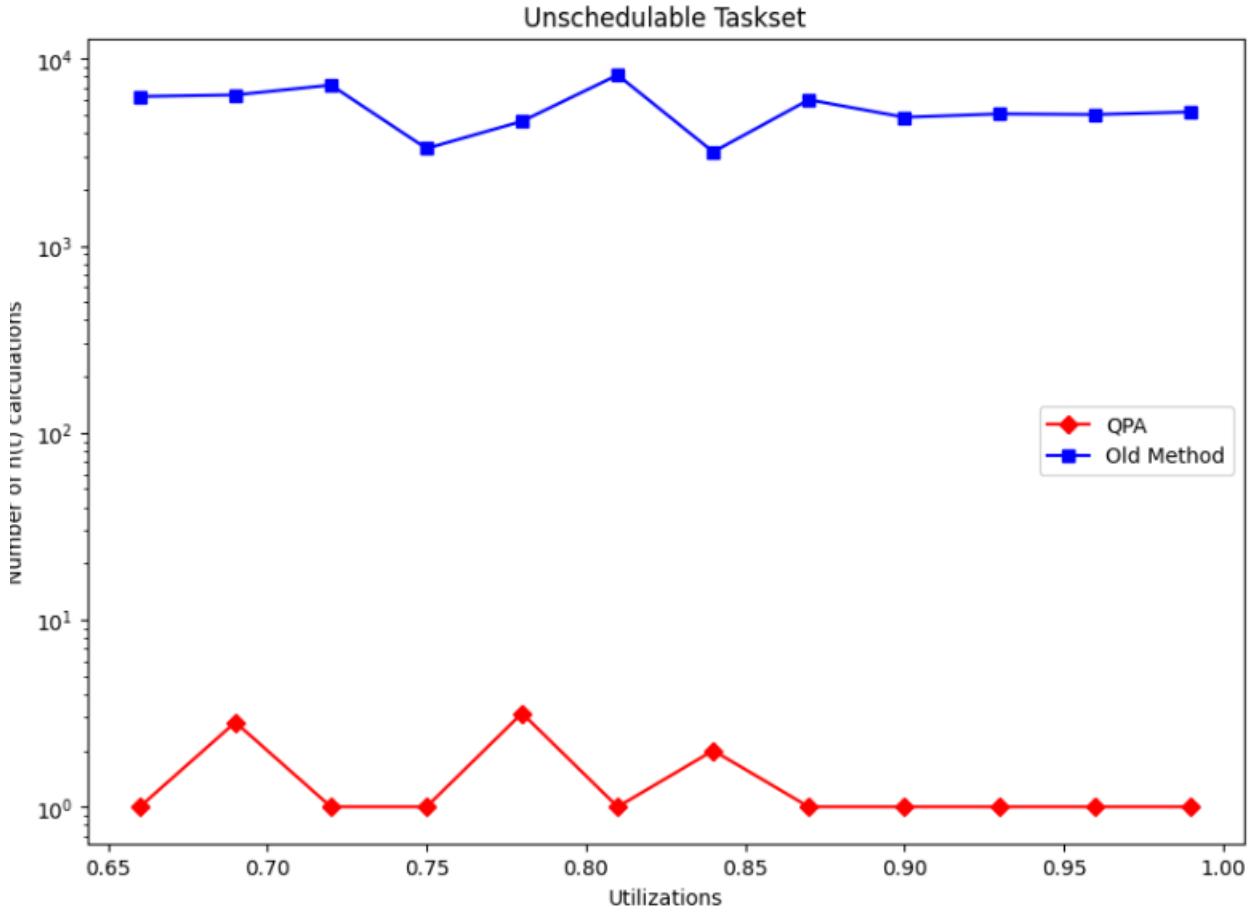




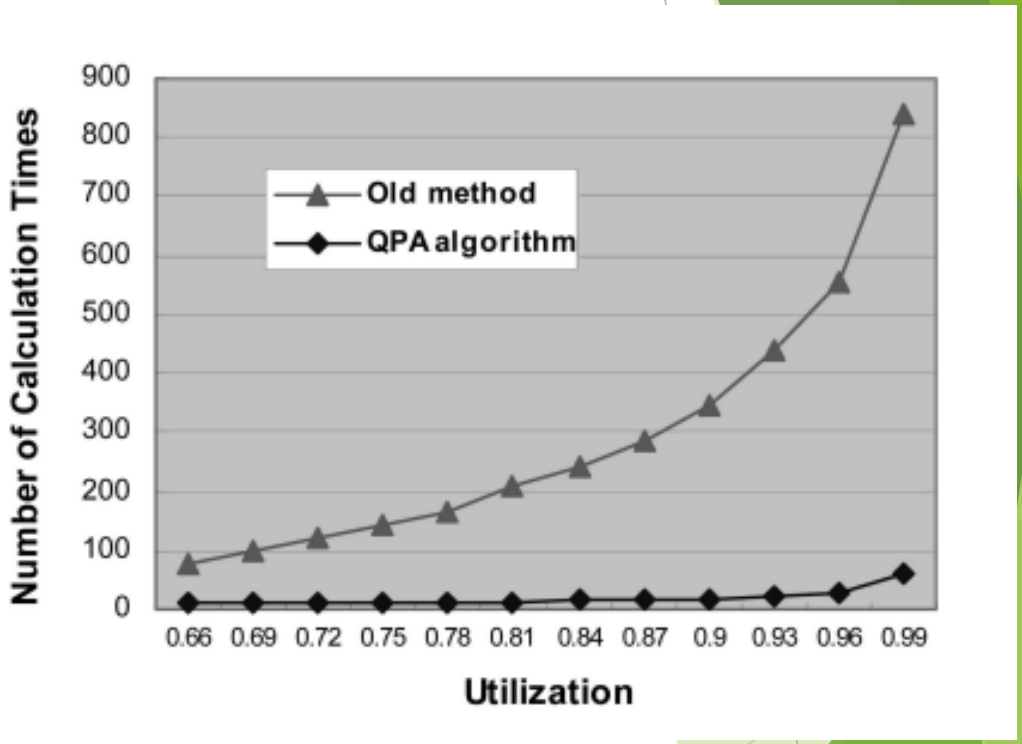
Impact of the Task Periods Range



number of tasks for each task set is 30,
utilization is 0.9.



Impact of the Utilization



The size of each task set is 30 and
Tmax=Tmin 1;000

Conclusion

- We have addressed and solved the problem of providing fast schedulability analysis which is necessary and sufficient for EDF scheduling with arbitrary relative deadlines.
- We can see that a number of factors can significantly affect the experimental results of the old methods; in some circumstances, they have exponential growth
- QPA, more than 96 percent of the task sets complete each schedulability test in less than 30 calculations of hðtÞ. The function hðtÞ has the complexity OðnÞ, equal to calculating a task set's utilization
- We also observed that the new upper bound L_a dominates L_b when each Di is no larger than $2T_i$. Since L_a^* is simpler to calculate than L_b, we would suggest that only L_a is used in QPA when each D_i is not significantly larger than $2T_i$.

References

- Reference for Lb bound
- Reference for h(t)

Critique

- Another state of the art algorithm for EDF schedulability analysis is the QPA improvement.
- QPA improvement does not calculate for every h(t). It is similar to binary search.
- It splits and tries to check for schedulability.

THANK YOU