Approximation scheme for real-time tasks under fixed-priority scheduling with deferred preemption

T.H.C. Nguyen\textsuperscript{1}, N.S. Tran\textsuperscript{1}, V.H. Le\textsuperscript{1}, P. Richard\textsuperscript{2}

\textsuperscript{(1)} VNU, University of Engineering and Technology, Hanoi, Vietnam  
\textsuperscript{(2)} LIAS, ENSMA and University of Poitiers, France
Outlines

1. **Problem statement**
   - Study background
   - Response time analysis: review
   - Studied problem

2. **Response time approximation**
   - Approximate functions
   - FPTAS principles
   - Numerical experiments
Motivations

Problem

Allowing arbitrary preemptions can introduce a high amount of runtime overhead.

Limiting preemptions in real-time tasks helps to:

- Improve I/O scheduling,
- Avoid mutual exclusion synchronizations
- Limit Cache Related Preemption delays (overhead due to cache misses,...)
Compromise between (arbitrary) preemptive and non-preemptive scheduling models:

- **Preemption thresholds**: disable preemption up to a specified priority level.
- **Floating Preemption**: maximum interval of non-preemptive regions for each task.
- **Deferred preemption** (Co-operative scheduling): Fixed preemption points (e.g., yield() call in the code).

Deferred Preemption Task Model

- Platform: Uniprocessor systems.
- Scheduler: Static-priority online scheduling algorithms.
- Task Sets: Sporadic tasks with arbitrary deadlines.
- Priorities: \( \text{Prio}(\tau_i) < \text{Prio}(\tau_j) \) iff \( i < j \).

Deferred Preemption Model:
- Every Job of \( \tau_i \) is a set of \( m_i \) non-preemptive subjobs.
- Preemptions are only allowed at subjob boundaries.
- Non-preemptive scheduling is a particular case.
Example with 3 subjobs: $\tau_i = \{s_{j1}, s_{j2}, s_{j3}\}$

- $C_i = \sum_{k=1}^{m_i} C_{i,k} = 7$: worst-case execution time of $\tau_i$
- $F_i = 3$: Computation time final subjob of $\tau_i$
- $D_i = 9$: Relative deadline of $\tau_i$
- $T_i = 10$: Minimum inter-arrival time of $\tau_i$
- $B_i$: Longest non-preemptive subjob among lower priority tasks
- $U_i = C_i / T_i$: Utilization factor
Existing Response Time Analysis

- Exact worst-case Response Time Analysis ($WR_i$)
  - Pseudo-polynomial time algorithm (Bril, et al, RTSJ. 2009)
  - NP-hard in the weak-sense (fixed-point computation)
  - No constant $c$ for approximation:
    \[ WR_i \leq \text{Approx}(WR_i) \leq c \times WR_i \]

- Response time upper bound
  - Linear bound (Davis, Burns, RTSS’08):
    \[
    \sup D(WR_i) = \frac{B_i + C_i - F_i + \sum_{j<i} (C_j(1 - U_j))}{1 - \sum_{j<i} U_j} + F_i
    \]
Existing Response Time Analysis

- Exact worst-case Response Time Analysis ($WR_i$)
  - Pseudo-polynomial time algorithm (Bril, et al, RTSJ. 2009)
  - NP-hard in the weak-sense (fixed-point computation)
  - No constant $c$ for approximation:
    \[ WR_i \leq \text{Approx}(WR_i) \leq c \times WR_i \]
- Response time upper bound
  - Linear bound (Davis, Burns, RTSS’08):
    \[
    \sup D(WR_i) = \frac{B_i + C_i - F_i + \sum_{j<i} (C_j(1 - U_j))}{1 - \sum_{j<i} U_j} + F_i
    \]
Existing Response Time Analysis

- Exact worst-case Response Time Analysis ($WR_i$)
  - Pseudo-polynomial time algorithm (Bril, et al, RTSJ. 2009)
  - NP-hard in the weak-sense (fixed-point computation)
  - No constant $c$ for approximation:
    $$WR_i \leq \text{Approx}(WR_i) \leq c \times WR_i$$

- Response time upper bound
  - Linear bound (Davis, Burns, RTSS’08):
    $$\sup D(WR_i) = \frac{B_i + C_i - F_i + \sum_{j<i} (C_j(1 - U_j))}{1 - \sum_{j<i} U_j} + F_i$$
This work

FPTAS (Fully Polynomial Time Approximation Scheme) : Response time upper bounds under resource augmentation.

- Parametric algorithm with input error $0 < \epsilon \leq 1$,
  $$k = \left\lceil \frac{1}{\epsilon} \right\rceil - 1.$$
- Let $WR_i$ be the exact worst-case response time upon a unit speed processor :
  - $WR_i \leq UB(WR_i)$ for a unit-speed processor.
  - $UB(WR_i) \leq WR_i$ for $(\frac{k}{k+1})$-speed processor.

This processor speedup is an upper bound on the price being paid for using an efficiently computable upper bound on response time!
This work

FPTAS (Fully Polynomial Time Approximation Scheme) : Response time upper bounds under resource augmentation.

- Parametric algorithm with input error $0 < \epsilon \leq 1$,
  
  \[ k = \left\lceil \frac{1}{\epsilon} \right\rceil - 1. \]

- Let $WR_i$ be the exact worst-case response time upon a unit speed processor :
  - $WR_i \leq UB(WR_i)$ for a unit-speed processor.
  - $UB(WR_i) \leq WR_i$ for $(\frac{k}{k+1})$-speed processor.

This processor speedup is an upper bound on the price being paid for using an efficiently computable upper bound on response time!
How to reduce the computational complexity for analysing $\tau_i$?

Level-$i$ active period: interval of time where only tasks with priority higher of equal to $\tau_i$ are running

<table>
<thead>
<tr>
<th>Exact Analysis</th>
<th>Approximate Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>solving fixed-point equations</td>
<td>intersection of two linear functions</td>
</tr>
<tr>
<td>Pseudo-polynomial number of $\tau_i$’s jobs in Level-$i$ active period</td>
<td>Polynomial number of $\tau_i$’s jobs Level-$i$ active period</td>
</tr>
</tbody>
</table>
Exact worst-case response time analysis

- Analysis of all jobs in the level-$i$ active period.
- Request bound functions in $[0, t)$ and $[0, t]$:

$$RBF(\tau_i, t) \overset{\text{def}}{=} \left\lceil \frac{t}{T_i} \right\rceil C_i$$
$$RBF'(\tau_i, t) \overset{\text{def}}{=} \left( \left\lfloor \frac{t}{T_i} \right\rfloor + 1 \right) C_i$$

- Cumulative workload functions of tasks having a priority higher or equal to $\tau_{i,l}$ plus a computation of length $C$:

$$wr_{i,l}(C, t) \overset{\text{def}}{=} C + (l + 1)C_i + \sum_{i<j} RBF(\tau_i, t)$$
$$wo_{i,l}(C, t) \overset{\text{def}}{=} C + (l + 1)C_i + \sum_{i<j} RBF'(\tau_i, t)$$
Exact worst-case response time analysis

- Analysis of all jobs in the level-\(i\) active period.
- Request bound functions in \([0, t)\) and \([0, t]\) :

\[
\text{RBF}(\tau_i, t) \overset{\text{def}}{=} \left\lceil \frac{t}{T_i} \right\rceil C_i \quad \text{RBF}'(\tau_i, t) \overset{\text{def}}{=} \left( \left\lfloor \frac{t}{T_i} \right\rfloor + 1 \right) C_i
\]

- Cumulative workload functions of tasks having a priority higher or equal to \(\tau_{i,l}\) plus a computation of length \(C\) :

\[
\text{wr}_{i,l}(C, t) \overset{\text{def}}{=} C + (l + 1)C_i + \sum_{i<j} \text{RBF}(\tau_i, t)
\]

\[
\text{wo}_{i,l}(C, t) \overset{\text{def}}{=} C + (l + 1)C_i + \sum_{i<j} \text{RBF}'(\tau_i, t)
\]
Approximate functions
FPTAS principles
Numerical experiments

Problem statement
Response time approximation
Conclusion and Perspectives

Exact worst-case response time analysis

Fixed point equations for the job $\tau_{i,l}$:

- Worst-case Response time $WR_{i,l}(C)$: smallest solution of $wr_{i,l}(C, t) = t$.
- Worst-case Occupied time $WO_{i,l}(C)$: smallest solution of $wo_{i,l}(C, t) = t$

Smallest fixed-point equations of $WR_{i,l}(C)$ and $WO_{i,l}(C)$ are used for computing the starting time of the final subjob of $\tau_{i,l}$:

$$R_{i,l} = \begin{cases} WR_{i,l}(B_i - F_i) & \text{for } i < n, \\ WO_{n,l}(-F_n) & \text{for } i = n. \end{cases}$$

Worst-case response time of $\tau_{i,l}$: $WR_{i,l} = R_{i,l} + F_i - l \times T_i$
Exact worst-case response time analysis

Fixed point equations for the job $\tau_{i,l}$:

- Worst-case Response time $WR_{i,l}(C)$: smallest solution of $wr_{i,l}(C, t) = t$.
- Worst-case Occupied time $WO_{i,l}(C)$: smallest solution of $wo_{i,l}(C, t) = t$

Smallest fixed-point equations of $WR_{i,l}(C)$ and $WO_{i,l}(C)$ are used for computing the starting time of the final subjob of $\tau_{i,l}$:

$$R_{i,l} = \begin{cases} 
WR_{i,l}(B_i - F_i) & \text{for } i < n, \\
WO_{n,l}(-F_n) & \text{for } i = n. 
\end{cases}$$

Worst-case response time of $\tau_{i,l}$: $WR_{i,l} = R_{i,l} + F_i - l \times T_i$
Fixed point equations for the job $\tau_{i,l}$:

- **Worst-case Response time** $WR_{i,l}(C)$: smallest solution of $wr_{i,l}(C, t) = t$.
- **Worst-case Occupied time** $WO_{i,l}(C)$: smallest solution of $wo_{i,l}(C, t) = t$

Smallest fixed-point equations of $WR_{i,l}(C)$ and $WO_{i,l}(C)$ are used for computing the \textit{starting time} of the final subjob of $\tau_{i,l}$:

$$R_{i,l} = \begin{cases} WR_{i,l}(B_i - F_i) & \text{for } i < n, \\ WO_{n,l}(-F_n) & \text{for } i = n. \end{cases}$$

Worst-case response time of $\tau_{i,l}$: $WR_{i,l} = R_{i,l} + F_i - l \times T_i$
Approximation scheme technique

$k$ : number of steps (scheduling points) to consider before the linear approximation.

\[
\overline{\text{RBF}}(\tau_i, t) = \begin{cases} 
\text{RBF}_i(t) & \text{for } t \leq (k - 1)T_i, \\
(t + T_i)\frac{c_i}{T_i} & \text{otherwise}.
\end{cases}
\]
Approximate starting time of final subjobs

Between two subsequent job releases, compute the intersection between:

- the processor capacity function $f(t) = t$
- approximate cumulative workload (linear)

![Graph showing the intersection of processor capacity function and approximate cumulative workload]
Approximate Workload Function and Testing Set

Scheduling Points (Testing set) :

\[ \hat{S}_i \overset{\text{def}}{=} \{ t = aT_b \mid a = 1, \ldots, k - 1; b = 1, \ldots, i - 1 \} \bigcup \{0\} \]

Let \( A \) denote the maximum instant in \( \hat{S}_i \) :

- \((0, A] : \forall j \leq i, \text{ approx. workload a step function.}\)
- \((A, \infty) : \forall j \leq i, \text{ approx. workload is a linear continuous function.}\)

\implies \text{corresponding to 2 testing stages.}
Stage 1: Primitive interval properties

There might be more than one job of $\tau_i$ to consider in a primitive interval $(t_1, t_2]$, but:

1. To check all jobs terminated against their deadlines: Check only the first job whose final subjob has started in $(t_1, t_2]$.

2. To check the end of the level-$i$ active period: Check if the last active period completes before the next job release.
Stage 1: Primitive interval properties

⇒ There might be more than one job of $\tau_i$ to consider in a primitive interval $(t_1, t_2]$, but:

1. To check all jobs terminated against their deadlines: Check only the first job whose final subjob has started in $(t_1, t_2]$.

2. To check the end of the level-\(i\) active period: Check if the last active period completes before the next job release.
Stage 1 : Primitive interval properties

There might be more than one job of $\tau_i$ to consider in a primitive interval $(t_1, t_2]$, but:

1. To check all jobs terminated against their deadlines: Check only the first job whose final subjob has started in $(t_1, t_2]$.

2. To check the end of the level-$i$ active period: Check if the last active period completes before the next job release.
Stage 1: Approximate Intersection Point

First Stage: Find an approximate intersection point in a primitive interval (two subsequent scheduling points).
Stage 2 : Linear approximation bound

Stage 2 analyses the primitive interval \((A, \infty)\) if level-\(i\) is not completed before the last scheduling point of the Stage 1.

- Define the index of the first job to complete in the interval \((A, \infty)\)
- Compute the intersection point between its approximate workload and the processor capacity

**Property**

The greatest upper bound computed during the two stages defines the approximate response time upper bound.
Stage 2 analyses the primitive interval \((A, \infty)\) if level-\(i\) is not completed before the last scheduling point of the Stage 1.

- Define the index of the first job to complete in the interval \((A, \infty)\)
- Compute the intersection point between its approximate workload and the processor capacity

**Property**

*The greatest upper bound computed during the two stages defines the approximate response time upper bound.*
Main properties of the algorithm:

- Performance guarantees:

**Lemma**

Let \( s = \frac{k}{k+1} \). If \((l + 1)C_i + C \geq 0\) then:

a. \( WR_{i,l}(C) \leq \hat{WR}_{i,l}(C) \leq WR_{i,s}(C) \).

b. \( WO_{i,l}(C) \leq \hat{WO}_{i,l}(C) \leq WO_{i,s}(C) \).

where \( k = \left\lceil \frac{1}{\epsilon} \right\rceil - 1 \)

- Worst-case speedup factor: \( (1 + \frac{k}{k+1}) \)
- Complexity of the algorithm: \( \mathcal{O}(kn^2) \) (This is an FTPAS)
Worst-case performance guarantee

Main properties of the algorithm:
- Performance guarantees:

**Lemma**

Let \( s = \frac{k}{k+1} \). If \((l + 1)C_i + C \geq 0\) then:

a. \( WR_{i,l}(C) \leq \widehat{WR}_{i,l}(C) \leq WR_{i,l}^{s}(C) \).

b. \( WO_{i,l}(C) \leq \widehat{WO}_{i,l}(C) \leq WO_{i,l}^{s}(C) \).

where \( k = \left\lceil \frac{1}{\epsilon} \right\rceil - 1 \)

- Worst-case speedup factor: \((1 + \frac{k}{k+1})\)

- Complexity of the algorithm: \(\mathcal{O}(kn^2)\) (This is an FTPAS)
Worst-case performance guarantee

Main properties of the algorithm:

- Performance guarantees:

### Lemma

Let $s = \frac{k}{k+1}$. If $(l + 1)C_i + C \geq 0$ then:

a. $WR_{i,l}(C) \leq \hat{WR}_{i,l}(C) \leq WR_{i,l}^s(C)$.

b. $WO_{i,l}(C) \leq \hat{WO}_{i,l}(C) \leq WO_{i,l}^s(C)$.

where $k = \lceil \frac{1}{\epsilon} \rceil - 1$

- Worst-case speedup factor: $(1 + \frac{k}{k+1})$

- Complexity of the algorithm: $O(kn^2)$ (This is an FTPAS)
Comparison of FPTAS and SupD (Davis, Burns, 2008) on randomly generated task sets.

Monitored metrics:

- Acceptance ratio: rate of tasks stated "feasible" by the considered upper bound ($ub_i \leq D_i$) for feasible tasks ($WR_i \leq D_i$).
- Average Error: $\frac{ub_i - WR_i}{WR_i}$

Plots:

- black: linear bound SupD (Davis, Burns 2008).
- others: our algorithm for $k = 2$ to $k = 5$. 
Acceptance ratio

![Graph showing acceptance ratio for different values of n and k]

- **Acceptance Ratio**
  - Y-axis: Acceptance Ratio (91% to 100%)
  - X-axis: n (10, 20, 30, 40, 50)
  - Legend:
    - SupD
    - k = 2
    - k = 3
    - k = 4
    - k = 5

The graph illustrates the acceptance ratio for various values of n and k, with SupD and different k values showing the trend in acceptance ratio.
Average Error

![Average Error Chart](image-url)

- **SupD**
- k = 2
- k = 3
- k = 4
- k = 5

- n values: 10, 20, 30, 40, 50
Approximate Worst-case Response Time analysis of FPDS:

- Polynomial Time Algorithm for response times upper bounds with high accuracy (FTPAS)
- Worst-case performance guarantee under resource augmentation analysis (i.e., speed up factor)

Perspectives:

- Release jitters and network analysis
- Analysis of large scale distributed systems under resource augmentation with a worst-case performance guarantee.
Conclusion and Perspectives

Approximate Worst-case Response Time analysis of FPDS:

- Polynomial Time Algorithm for response times upper bounds with high accuracy (FTPAS)
- Worst-case performance guarantee under resource augmentation analysis (i.e., speed up factor)

Perspectives:

- Release jitters and network analysis
- Analysis of large scale distributed systems under resource augmentation with a worst-case performance guarantee.