A bicriteria PTAS for Real-time Scheduling with fixed priorities

Thomas Rothvoß
University of Paderborn

8.01.08

This is joint work with Fritz Eisenbrand
Content of the talk

1. Preliminaries

2. Rounding the instance

3. The algorithm for “large” tasks

4. Dealing with “small” tasks

5. Open problems
Definition

Problem: Given periodic Tasks $T_1, \ldots, T_n$ with implicit deadlines such that each $T_i$ has running time $c_i$ and period $p_i$. Task $T_i$ generates a job of running time $c_i$ each $p_i$ time units, that has to be completed before its period ends.

Goal: Distribute tasks among as few processors as possible using preemptive scheduling.

![Diagram of task scheduling](image)
Dynamic priorities

Theorem. Dynamic priorities & preemptive Scheduling: Earliest-Deadline First is optimal

Def.: \( \frac{c_i}{p_i} = \text{utilization of } T_i \)

Theorem. [Liu, Layland ’73] If dynamic priorities are allowed: Tasks are feasible on a single processor \( \iff \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1 \)

\( \Rightarrow \) Bin Packing with item sizes \( \frac{c_i}{p_i} \) and bin size 1.
Fixed priorities

Theorem. [Liu, Layland ’73] Optimal priorities: $\frac{1}{p_i}$ for $T_i$ (Rate-monotonic Schedule)

Example

\[
\begin{align*}
    c_1 &= 1, \quad p_1 = 2 \\
    c_2 &= 2, \quad p_2 = 5
\end{align*}
\]

---

\begin{tikzpicture}
    \draw[->] (0,0) -- (10,0) node[right] {10 \quad time};
    \draw (0,0) -- (0,1.5);
    \foreach \x in {1,2,3,4,5,6,7,8,9,10}
    \draw (\x,0.1) -- (\x,-0.1) node[below] {$\x$};
    \draw (0,1) -- (10,1) node[right] {time};
\end{tikzpicture}
Fixed priorities

**Theorem.** [Liu, Layland ’73] Optimal priorities: \( \frac{1}{p_i} \) for \( T_i \) (Rate-monotonic Schedule)

**Example**

\[
\begin{align*}
c_1 &= 1, \ p_1 = 2 \\
c_2 &= 2, \ p_2 = 5
\end{align*}
\]
Fixed priorities

Theorem. [Liu, Layland ’73] Optimal priorities: \( \frac{1}{p_i} \) for \( T_i \) (Rate-monotonic Schedule)

Example

c_1 = 1, p_1 = 2

c_2 = 2, p_2 = 5
Fixed priorities

**Theorem.** [Liu, Layland ’73] Optimal priorities: $\frac{1}{p_i}$ for $T_i$ (Rate-monotonic Schedule)

**Example**

\[c_1 = 1, \ p_1 = 2\]
\[c_2 = 2, \ p_2 = 5\]

Utilization $\frac{1}{2} + \frac{2}{5} = \frac{9}{10}$
Feasibility

Lemma. [Liu, Layland ’73] \( \sum_{i=1}^{n} \frac{c_i}{p_i} \leq \ln(2) \approx 0.69 \Rightarrow \) tasks feasible on a single processor

Def.: \( r_i = \) be Response Time of \( T_i = \) longest time that an instance of task \( T_i \) waits for accomplishment

Lemma. [Lehoczky et al. ’89] If \( p_1 \leq \ldots \leq p_n \) then \( r_i \) is the smallest value s.t.

\[
r_i = c_i + \sum_{j<i} \left\lceil \frac{r_i}{p_j} \right\rceil c_j
\]

Tasks are feasible \( \Leftrightarrow \forall i : r_i \leq p_i \)
Local feasibility

Def.: Task $T_i$ is locally feasible if $\exists r_i \leq p_i$

$$c_i + \sum_{j<i} \left\lceil \frac{r_i}{p_j} \right\rceil c_j \leq r_i$$
Local feasibility

Def.: Task $T_i$ is \textit{locally feasible} if $\exists r_i \leq p_i$

\[
c_i + \sum_{j<i, p_j \leq \varepsilon p_i} \left\lfloor \frac{r_i}{p_j} \right\rfloor c_j + \sum_{j<i, p_j > \varepsilon p_i} \left\lfloor \frac{r_i}{p_j} \right\rfloor c_j \leq r_i
\]
Local feasibility

Def.: Task $T_i$ is *locally feasible* if $\exists r_i \leq p_i$

$$c_i + \sum_{j<i, p_j \leq \varepsilon p_i} \frac{r_i}{p_j} c_j + \sum_{j<i, p_j > \varepsilon p_i} \left\lfloor \frac{r_i}{p_j} \right\rfloor c_j \leq r_i$$
Local feasibility

Def.: Task $T_i$ is *locally feasible* if $\exists r_i \leq p_i$

$$c_i + r_i \cdot \sum_{j < i, p_j \leq \epsilon p_i} \frac{c_j}{p_j} + \sum_{j < i, p_j > \epsilon p_i} \left\lceil \frac{r_i}{p_j} \right\rceil c_j \leq r_i$$

Lemma. *Tasks locally feasible w.r.t. $\epsilon > 0$ ⇒ tasks feasible on a processor of speed $1 + 2\epsilon$*

Our main result

**Theorem.** *For any $\epsilon > 0$ we can schedule tasks on $(1 + \epsilon)OPT + O(1)$ many processors with speed $1 + \epsilon$ in polynomial time.*

Recently best algorithm: $\frac{7}{4}$-approximation [Burchard et al. ’95]
Rounding the instance

Assume that $\frac{c_i}{p_i} \geq \varepsilon$ and $\frac{1}{\varepsilon} \in \mathbb{Z}$. Round such that

- $p_i = (1 + \varepsilon)^Z$
- $\frac{c_i}{p_i} \in \{0, \varepsilon^2, 2\varepsilon^2, ..., 1\}$
- Choose $k \in \{0, ..., 1/\varepsilon - 1\}$ randomly. Remove all tasks having their period in an interval $[1/\varepsilon^i, 1/\varepsilon^{i+1}]$ with $i \equiv 1/\varepsilon \cdot k$. 

period
Rounding the instance

Assume that \( \frac{c_i}{p_i} \geq \varepsilon \) and \( \frac{1}{\varepsilon} \in \mathbb{Z} \). Round such that

- \( p_i = (1 + \varepsilon)^Z \)
- \( \frac{c_i}{p_i} \in \{0, \varepsilon^2, 2\varepsilon^2, ..., 1\} \)
- Choose \( k \in \{0, ..., 1/\varepsilon - 1\} \) randomly. Remove all tasks having their period in an interval \([1/\varepsilon^i, 1/\varepsilon^{i+1}[^{\varepsilon} \) with \( i \equiv 1/\varepsilon \, k \).
Rounding the instance

Assume that $\frac{c_i}{p_i} \geq \varepsilon$ and $\frac{1}{\varepsilon} \in \mathbb{Z}$. Round such that

- $p_i = (1 + \varepsilon)^Z$
- $\frac{c_i}{p_i} \in \{0, \varepsilon^2, 2\varepsilon^2, ..., 1\}$
- Choose $k \in \{0, ..., 1/\varepsilon - 1\}$ randomly. Remove all tasks having their period in an interval $[1/\varepsilon^i, 1/\varepsilon^{i+1}]$ with $i \equiv 1/\varepsilon \ k$. 

![Diagram showing intervals and periods](image-url)
Rounding the instance

Assume that \( \frac{c_i}{p_i} \geq \varepsilon \) and \( \frac{1}{\varepsilon} \in \mathbb{Z} \). Round such that

- \( p_i = (1 + \varepsilon)^z \)

- \( \frac{c_i}{p_i} \in \{0, \varepsilon^2, 2\varepsilon^2, \ldots, 1\} \)

- Choose \( k \in \{0, \ldots, 1/\varepsilon - 1\} \) randomly. Remove all tasks having their period in an interval \([1/\varepsilon^i, 1/\varepsilon^{i+1}]\) with \( i \equiv 1/\varepsilon \ k \).
Rounding the instance

Assume that \( \frac{c_i}{p_i} \geq \varepsilon \) and \( \frac{1}{\varepsilon} \in \mathbb{Z} \). Round such that

- \( p_i = (1 + \varepsilon)^Z \)

- \( \frac{c_i}{p_i} \in \{0, \varepsilon^2, 2\varepsilon^2, ..., 1\} \)

- Choose \( k \in \{0, ..., 1/\varepsilon - 1\} \) randomly. Remove all tasks having their period in an interval \([1/\varepsilon^i, 1/\varepsilon^{i+1}]\) with \( i \equiv 1/\varepsilon \, k \).

\[ \Rightarrow \text{Blocks } B_1, ..., B_k \]
Dynamic programming

\[ A(a_1, \ldots, a_n, \ell) = \begin{cases} 
1 & \text{if tasks in } B_1, \ldots, B_\ell \text{ can locally feasible distributed s.t. processor } i \text{ has util. } \leq a_i \\
0 & \text{otherwise} 
\end{cases} \]

Compute

\[ A(a_1, \ldots, a_n, \ell) = 1 \iff \exists 0 \leq b_i \leq a_i : A(b_1, \ldots, b_n, \ell - 1) \& \text{tasks } B_\ell \text{ can distributed feasibly} \]

Finally we need

\[ \min \{ j \mid A(1, \ldots, 1, 0, \ldots, 0, k) = 1 \} \]

many processors.
Distribution of $B_\ell$

- # of utilization values: $\frac{1}{\varepsilon^2} + 1 = O(1)$

- # of periods in $B_\ell$: $1 + \log_{1+\varepsilon}(1/\varepsilon)^{1/\varepsilon} - 1 = O(1)$

- $\Rightarrow$ # of different task types in $B_\ell$: $O(1)$

- Utilization $\geq \varepsilon \Rightarrow \leq \frac{1}{\varepsilon}$ tasks per processor

- $\Rightarrow O(1)$ possible packings

- Utilization on processor $i$ can be increased from $b_i \in \varepsilon^2\mathbb{Z}$ to $a_i \in \varepsilon^2\mathbb{Z} \Rightarrow O(1)$

- $n^{O(1)}$ many ways to distribute tasks in $B_\ell$ among the processors
Dealing with small tasks

Partition tasks with $\text{util} \leq \varepsilon^6$ in $R_1 \cup \ldots \cup R_m$ such that

- all tasks in $R_i$ have the same period.
- $\varepsilon^6 \leq \text{utilization of } R_i \leq 3\varepsilon^6$

Glue tasks in each $R_i$ together
Merging theorem

**Theorem.** \( \mathcal{I}' \) merged instance \( \Rightarrow \exists \) solution for \( \mathcal{I}' \) with \( (1 + \varepsilon)OPT + O(1) \) processors of speed \( 1 + \varepsilon \)

Let \( S_1 \cup \ldots \cup S_m \) optimal solution for \( \mathcal{I} \). Consider group \( R_j \) choose a task \( T \in R_j \) randomly with prob \( \frac{\text{utilization of } T}{\text{utilization of } R_j} \). Put new task for \( R_j \) on \( T \)'s processor.

\[ \text{tasks in } R_j \]

proc. 1 \hspace{0.5cm} proc. 2 \hspace{0.5cm} \ldots \hspace{0.5cm} \text{proc. } m \]
**Theorem.** \( \mathcal{I}' \) merged instance \( \Rightarrow \exists \) solution for \( \mathcal{I}' \) with \((1 + \varepsilon)\text{OPT} + O(1)\) processors of speed \(1 + \varepsilon\)

Let \( S_1 \dot{\cup} \ldots \dot{\cup} S_m \) optimal solution for \( \mathcal{I} \). Consider group \( R_j \) choose a task \( T \in R_j \) randomly with prob \( \frac{\text{utilization of } T}{\text{utilization of } R_j} \). Put new task for \( R_j \) on \( T \)'s processor.

![Diagram of processors and tasks](image-url)
Merging theorem

**Theorem.** $\mathcal{I}'$ merged instance $\Rightarrow \exists$ solution for $\mathcal{I}'$ with $(1 + \varepsilon)OPT + O(1)$ processors of speed $1 + \varepsilon$

Let $S_1 \cup \ldots \cup S_m$ optimal solution for $\mathcal{I}$. Consider group $R_j$ choose a task $T \in R_j$ randomly with prob $\frac{\text{utilization of } T}{\text{utilization of } R_j}$. Put new task for $R_j$ on $T$'s processor.
Merging theorem

Task $T_i$ still feasible on a processor of speed $1 + \varepsilon \Leftrightarrow \exists r_i \leq p_i$:

$$c_i + \sum_{j < i, T_i \text{ large}} \left\lfloor \frac{r_i}{p_j} \right\rfloor c_j + r_i \sum_{j < i, T_i \text{ small}} \left\lfloor \frac{r_i}{p_j} \right\rfloor \frac{p_j}{r_i} \in [1,2] \text{ utilization} \leq (1 + \varepsilon)r_i$$

Via Chernoff bounds: $\Pr[T_i \text{ is not feasible}] \leq \varepsilon$

If $T_i$ gets infeasible $\Rightarrow$ remove $T_i$
Open problems

- What about a real (asymptotic) PTAS?

- Now running time $n^{g(\varepsilon)}$. Is a bicriteria FPTAS possible or at least running time $f(\varepsilon) \cdot n^{O(1)}$?

- Absolutely inefficient in practice! Is there a practicable algorithm (better than First-Fit)?