Scheduling Aperiodic Jobs on Uniprocessor Systems

Prof. Dr. Jian-Jia Chen

LS 12, TU Dortmund

25 April 2016
A job $J$ is characterized as follows:

- Arrival time ($a_j$) or release time ($r_j$) is the time at which the job becomes ready for execution.
- Computation (execution) time ($C_j$) is the time necessary to the processor for executing the job without interruption (= WCET).
- Absolute deadline ($d_j$) is the time at which the job should be completed.
Earliest Due Date Algorithm

Theorem

Given a set of $n$ independent jobs that arrive synchronously (release time is 0), any algorithm that executes tasks in order of non-decreasing absolute deadlines is optimal with respect to minimizing the maximum lateness.

Denoted as Earliest Due Date (EDD) Algorithm [Jackson, 1955]

Proof

Let $\sigma$ be the schedule for $J$ produced by scheduling algorithm $A$. We can transform $A$ to EDD schedule $A'$ without increasing $L_{\text{max}}$. Details are in the textbook by Buttazzo [Theorem 3.1].
A Sketched Proof

EDD is optimal for minimizing the maximum lateness: Given a set of n independent tasks, any algorithm that executes the tasks in order of non-decreasing (absolute) deadlines is optimal with respect to minimizing the maximum lateness.

Proof (Buttazzo, 2002):

- Let $S$ be a schedule produced by any algorithm $A$
- If $S \neq$ the schedule of $EDD$
  - $\exists J_a, J_b : d_a \leq d_b$, $J_b$ immediately precedes $J_a$ in $S$.
- Let $S'$ be the schedule obtained by exchanging $J_a$ and $J_b$
A Sketched Proof (contd.)

Exchanging $J_a$ and $J_b$ cannot increase lateness

Max. lateness for $J_a$ and $J_b$ in $S$ is $L_{\text{max}}(a, b) = f_a - d_a$

Max. lateness for $J_a$ and $J_b$ in $S'$ is $L_{\text{max}}(a, b) = \max(L'_a, L'_b)$

Two possible cases:

1. $L'_a \geq L'_b \implies L'_{\text{max}}(a, b) = f'_a - d_a < f_a - d_a = L_{\text{max}}(a, b)$ since $J_a$ starts earlier in schedule $S'$.

2. $L'_a < L'_b \implies L'_{\text{max}}(a, b) = f'_a - d_a = f_a - d_b \leq f_a - d_a = L_{\text{max}}(a, b)$ since $f_a = f'_b$ and $d_a \leq D_b$

$\implies L'_{\text{max}}(a, b) \leq L_{\text{max}}(a, b)$
Optimality of EDF

**Theorem**

Given a set of $n$ independent aperiodic tasks (jobs) with arbitrary arrival times, if the aperiodic task set is feasible on a single processor then any algorithm that executes tasks with earliest deadline (among the set of active tasks) is guaranteed to meet all tasks’ deadlines.

- Similar to Jackson Algorithm proof of optimality, but need to account for preemption. (The steps are almost the same as the previous slide, by accounting for preemptions. I leave this proof out.)
Exact Schedulability Test for EDF

**Theorem**

A set of aperiodic tasks is schedulable (by EDF) if and only if

\[
\forall a_i < d_k, \quad \sum_{\tau_j: a_i \leq a_j \text{ and } d_j \leq d_k} C_j \leq d_k - a_i
\]

- **Proof for only if** (necessary test): this simply comes from the fact that the demand must be no more than the available time

- **Now**: Proof for if (sufficient test):
  use contrapositive: \( A \rightarrow B \iff \overline{B} \rightarrow \overline{A} \)
Proof: Sufficient Schedulability Test for EDF

Proof by contrapositive:

- Suppose that EDF schedule does not meet the deadline
- Let $\tau_k$ be the first task which misses its absolute deadline $d_k$
- Let $t_0$ be the last instant before $d_k$, at which either the processor is idle or the processor executes a task with absolute deadline larger than $d_k$
- By EDF, $t_0$ must be an arrival time of a job, called $\tau_i$
- Therefore, $t_0$ is equal to $a_i$
- The processor executes only the jobs arriving no earlier than $a_i$ and with absolute deadline less than or equal to $d_k$
- Therefore, we conclude the proof by showing that
  \[
  \exists a_i < d_k, \sum_{\tau_j: a_i \leq a_j \text{ and } d_j \leq d_k} C_j > d_k - a_i
  \]
Theorem

Define demand bound function $dbf(\tau_i, t)$ as

$$dbf(\tau_i, t) = \max \left\{ 0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \right\}$$

$$C_i = \max \left\{ 0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right\}$$

A task set $\mathcal{T}$ of independent, preemptable, periodic tasks can be feasibly scheduled (under EDF) on one processor if and only if

$$\forall \ L \geq 0, \ \sum_{i=1}^{n} dbf(\tau_i, L) \leq L.$$
Least Laxity First

Priorities = decreasing function of the laxity (lower laxity = higher priority); changing priority; preemptive.

\[
\begin{array}{|c|c|c|c|}
\hline
\tau_1 & \tau_2 & \tau_3 \\
\hline
a_j & 0 & 4 & 5 \\
C_j & 10 & 3 & 10 \\
d_j & 33 & 28 & 29 \\
\hline
\end{array}
\]

\[
\begin{align*}
l(\tau_1) &= 33 - 4 - 6 = 23 \\
l(\tau_2) &= 28 - 4 - 3 = 21 \\
l(\tau_1) &= 33 - 15 - 6 = 12 \\
l(\tau_2) &= 28 - 15 - 2 = 13 \\
l(\tau_3) &= 29 - 15 - 2 = 12 \\
l(\tau_1) &= 33 - 16 - 6 = 11 \\
l(\tau_2) &= 28 - 16 - 1 = 12 \\
l(\tau_3) &= 29 - 16 - 2 = 14 \\
\end{align*}
\]