Priority-Driven Scheduling of Periodic Tasks (2)
- Chapter 6 -

Schedulable utilization bound

- Simpler method for the schedulabiity check

Utilization

- A periodic task's utilization U_i of an active resource is the ratio between its execution time and period: $U_i = C_i/p_i$
- Given a set of periodic tasks on an active resource, e.g. the CPU, the CPU's utilization is equal to the sum of periodic tasks' utilization:

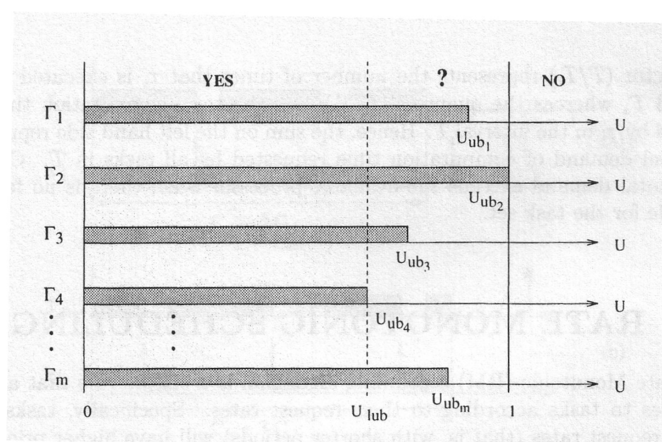
$$U = \sum_i \frac{C_i}{p_i}$$

- Can we find a bound called "schedulable utilization bound" under which a task set is guaranteed to be schedulable?

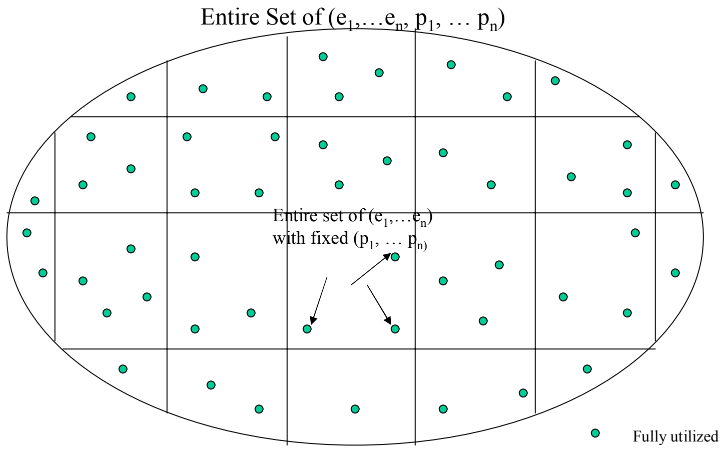
$$\text{if } U = \sum_i \frac{C_i}{p_i} \le U_{bound}\text{, task set is schedulable}$$

Processor utilization factor

- For a given algorithm A, we are interested in finding its schedulability bound (e.g., the schedulability bound of EDF is 1)



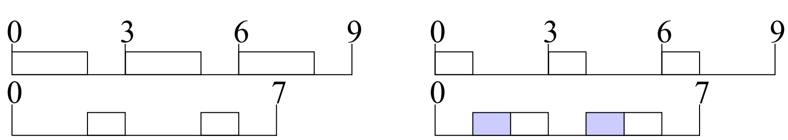
Processor utilization factor



Find the minimum utilization factor among all fully utilized dots
(barelly schedulable task set)

Which pattern of *e* and *p* values?

- Now, we can consider only (e and p) combinations that make the system barely schedulable.
- How to find a (e and p) combination that has the minimal utilization factor?
- Always start with examples→ intuition→ generic theorem



$U = 2/3 + 2/7$

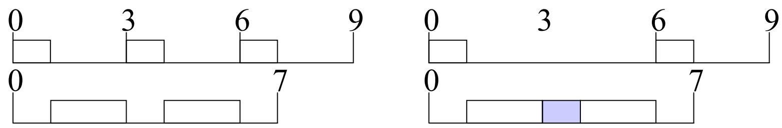
$U = (2-\Delta)/3 + (2+2\Delta)/7$

decrease : $\Delta/3$

increase : $\Delta\lfloor 7/3 \rfloor/7 < \Delta(7/3)/7 = \Delta/3$

Which pattern of *p* values?

- For e values: “No-overflow theorem”
- What about p values?



U = 1/3 + 4/7

U = 1/(3*2) + (4+1)/7

Decrease: 1/3 – 1/(2*3)=1/(2*3)

Increase: 1/7

Since 7 > 2*3, U decreases

Transform (Ratio 3) to 2

$\left\lceil \frac{p_2}{p_1} \right\rceil e_1 + e_2 = p_2$; $\quad$ where $\left\lceil \frac{p_2}{p_1} \right\rceil = 3$, e.g. $\left\lceil \frac{10}{4} \right\rceil 2 + 4 = 10$

$\left\lceil \frac{p_2}{2p_1} \right\rceil e_1 + e_2 + e_1 = p_2$, if $\left\lceil \frac{p_2}{2p_1} \right\rceil = 2$ $\quad$ $\left\lceil \frac{10}{2*4} \right\rceil 2 + 4 + 2 = 10$

} p_1 is doubled, so we obtain **ratio 2** among the periods

$(\frac{e_1}{p_1} + \frac{e_2}{p_2}) - (\frac{e_1}{2p_1} + \frac{e_2 + e_1}{p_2})$ $\quad$ $(\frac{2}{4} + \frac{4}{10}) - (\frac{2}{2*4} + \frac{4+2}{10}) > 0$

$= (\frac{e_1}{2p_1} - \frac{e_1}{p_2}) > 0$, if $p_2 > 2p_1$

$Since \left\lceil \frac{p_2}{p_1} \right\rceil = 3$, we have $3 > \frac{p_2}{p_1} > 2$. Thus, $1.5 > \frac{p_2}{2p_1} > 1$ and $\left\lceil \frac{p_2}{2p_1} \right\rceil = 2$; $p_2 > 2p_1$

Transform (Ratio k > 2) to 2

$\left\lceil \frac{p_2}{p_1} \right\rceil = k$, we have $k > \frac{p_2}{p_1} > k-1$. Thus, $\frac{k}{k-1} > \frac{p_2}{(k-1)p_1} > 1$ and $\left\lceil \frac{p_2}{(k-1)p_1} \right\rceil = 2$

$\left\lceil \frac{p_2}{p_1} \right\rceil e_1 + e_2 = p_2$; original

$\left\lceil \frac{p_2}{(k-1)p_1} \right\rceil e_1 + e_2 + (k-2)e_1 = 2e_1 + e_2 + (k-2)e_1 = p_2$; after transform

$$\left(\frac{e_1}{p_1} + \frac{e_2}{p_2}\right) - \left(\frac{e_1}{(k-1)p_1} + \frac{e_2 + (k-2)e_1}{p_2}\right)$$

$$= \left(\frac{(k-2)e_1}{(k-1)p_1} + \frac{(k-2)e_1}{p_2}\right) > 0, \text{since } (k-1)p_1 < p_2$$

The Remaining Free Variables

- The computation time for each task is as follows

$$e_1 = p_2 - p_1; e_2 = p_3 - p_2; \ldots; e_{n-1} = p_n - p_{n-1}$$

$$\begin{aligned} e_n &= p_n - 2(e_1 + e_2 + \ldots + e_{n-1}) \\ &= p_n - 2(p_2 - p_1 + p_3 - p_2 + \ldots + p_n - p_{n-1}) \\ &= 2p_1 - p_n \end{aligned}$$

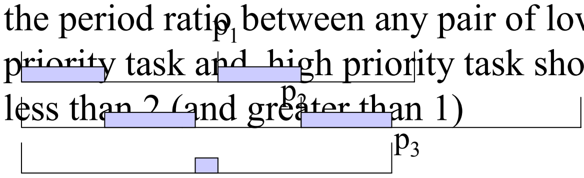
- Quiz: what are the remaining free variables?

  what the type of the variables?

  what is the standard tool to get a maximum or minimal?

Putting Things Together

- The tasks' pattern that leads to minimal utilization is
  - the execution time shall not overflow
  - the period ratio between any pair of low priority task and high priority task should be less than 2 (and greater than 1)



Solving the PDE

$$U = \frac{e_1}{p_1} + \ldots + \frac{e_n}{p_n} = \frac{p_2 - p_1}{p_1} + \ldots + \frac{p_n - p_{n-1}}{p_{n-1}} + \frac{2p_1 - p_n}{p_n}$$

$$= \frac{p_2}{p_1} + \frac{p_3}{p_2} + \ldots + \frac{p_n}{p_{n-1}} + \frac{2p_1 p_2 \ldots p_{n-1}}{p_2 \ldots p_{n-1} p_n} - n$$

$$= r_1 + r_2 + \ldots + r_{n-1} + \frac{2}{r_1 r_2 \ldots r_{n-1}} - n; \quad where \quad r_i = \frac{p_{i+1}}{p_i}$$

$$set \frac{\partial U}{\partial r_i} = 0; \; we\, have$$

$$r_1^2 r_2 \ldots r_{n-1} = 2 \quad (1); \; r_1 r_2^2 \ldots r_{n-1} = 2 \quad (2); \ldots; \; r_1 r_2 \ldots r_{n-1}^2 = 2 \quad (n-1)$$

$(1)/(2) = 1 \;\; => \; r_1 = r_2$

Dividing them successively, we have $r_1 = r_2 = \ldots = r_{n-1}$

Plug it back to (1) we have $r = 2^{1/n}$

$$U = (n-1)2^{1/n} + \frac{2}{2^{(n-1)/n}} - n = n(2^{1/n} - 1)$$

The L&L Bound

A set of n periodic task is schedulable if :

$$\frac{e_1}{p_1} + \frac{e_2}{p_2} + \ldots + \frac{e_n}{p_n} \le n(2^{1/n} - 1)$$

• U(1) = 1.0	U(4) = 0.756	U(7) = 0.728
• U(2) = 0.828	U(5) = 0.743	U(8) = 0.724
• U(3) = 0.779	U(6) = 0.734	U(9) = 0.720

- For harmonic task sets, the utilization bound is U(n)=1.00 for all n. For large n, the bound converges to *ln 2* ~ 0.69.
- The L&L bound for rate monotonic algorithm is one of the most significant results in real-time scheduling theory. Its derivation also shows a wealth of analysis techniques that are useful in many new situations when considering static priority scheduling.

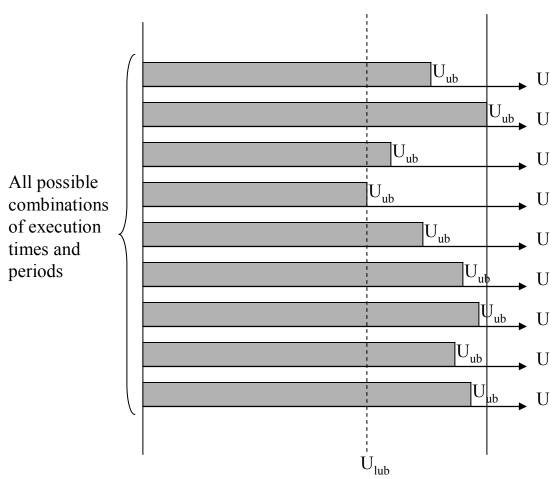
Summary of Utilization Bound

- The minimum utilization factor among all barely schedulable task sets is a sufficient bound for the schedulability
- One time check with simple comparison
- Still sufficient condition
  - even if task phase never make critical instant
  - execution times are smaller than the given values
  - inter-release time is longer than the given periods
- Problems
  - Only sufficient condition
  - we cannot say anything if utilization is higher than the bound – safe choice is to assume it is not schedulable

Enhancement of Utilization Bound

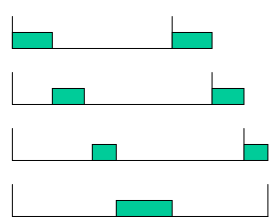
- L&L bound takes the worst case (with minimum utilization factor) worrying about all possible values of execution times and periods
- If some parameters are fixed, L&L worst case may not happen, and hence L&L bound is unnecessarily pessimistic for such limited problem scope
- What if we know period values?
- The more we know the higher is the schedulability bound.

L&L Bound (Review)

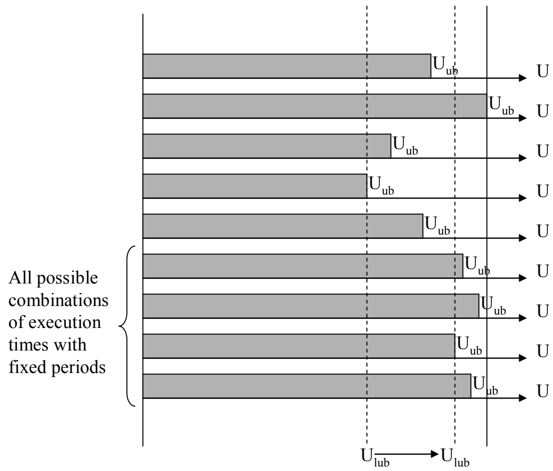


U_{lub} can be found when

$$p_1 \le p_2 \le \cdots \le p_k \le 2p_1,$$



The tight bound when period values are fixed

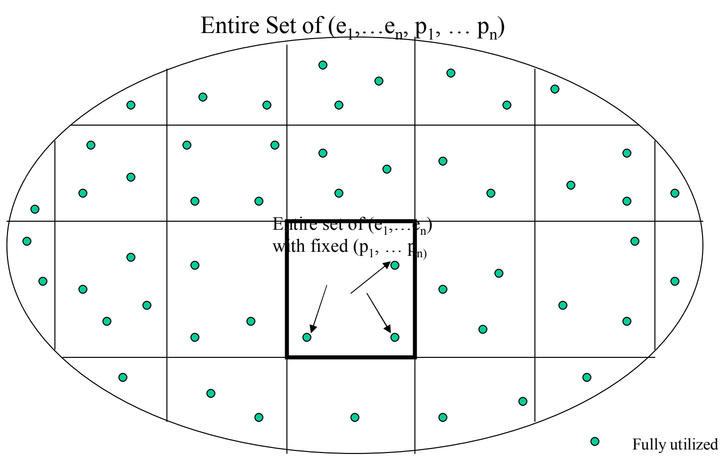


The previous condition cannot give the U_{lub} when period values are fixed.

The given period values may not meet $p_1 \le p_2 \le \cdots \le p_k \le 2p_1$.

In such case, no simple relation exists among execution times for U_{lub}.

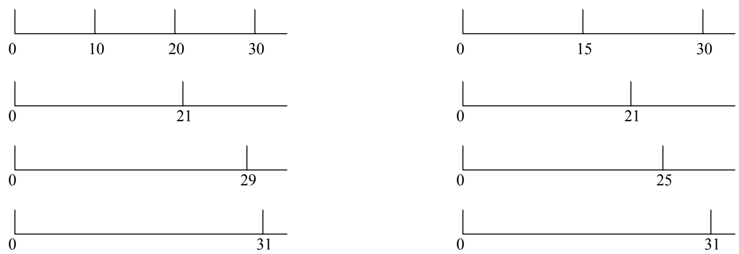
We know period values $(p_1, p_2, \ldots, p_n)$



Find the minimum utilization factor within the limited scope

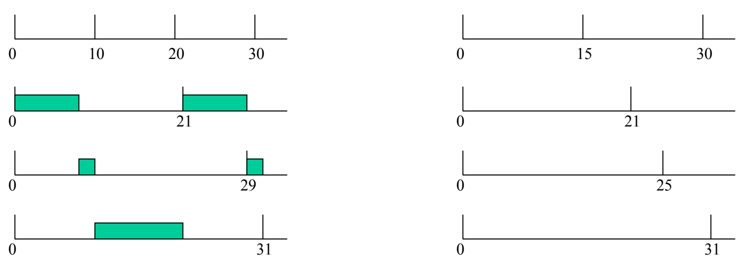
Examples

(Execution time alloc. for fixed period values)



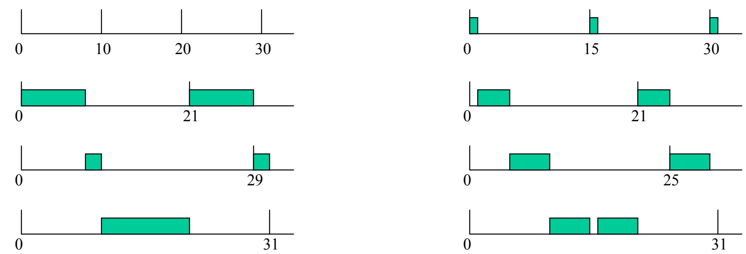
Examples

(Execution time alloc. for fixed period values)



Examples

(Execution time alloc. for fixed period values)



Optimization Formulation

- Actually, minimization problem

$$\text{Minimize} \sum_{j=1}^{n} \frac{e_j}{p_j}$$

Subject to

"Barely schedulable"

where $p_j (1 \le j \le K)$ are fixed values

and $e_j (1 \le j \le K)$ are free variables

Use Linear Programming

- Level-i bound U_i^{bound}: only guarantees the schedulability of task i

$$U_i^{bound} = \text{minimize} \sum_{j=1}^{i} \frac{e_j}{p_j}$$

subject to

$$\sum_{j=1}^{i-1} \left\lceil \frac{p_i}{p_j} \right\rceil e_j + e_i \le p_i, \quad \text{// make task } i \text{ schedulable}$$

$$\sum_{j=1}^{i-1} \left\lceil \frac{t_a}{p_j} \right\rceil e_j + e_i \ge t_a \text{ for all } t_a (1 \le a \le M) \quad \text{// make } [0, p_i] \text{ fully utilized}$$

where $t_a (1 \le a \le M)$ are the series of all the release times of higher priority tasks in $[0, p_i]$.

- System-level bound U^{bound}: guarantees the schedulability of the task set

$$U^{bound} = \min_{i=1}^{n} U_i^{bound}$$

LP_SOLVER

- Problem description (inFile)

```
min: 0.35 x1 + 2.03 x2;
2.01 x1 + 0.32 x2 = 120.0;
-4.0 x1 + 3.3 x2 <= 5.0;

int x2, x1;
```

- Run lp_solver

```
lp_solver < inFile > outFile
```

- Output file (outFile)

```
Value of objective function: 0.8599
x1        20
x2        15
```

Practical Issues

- Practical Issues
  - What if there is a non-preemptable code section (e.g., system call)?
  - What if the context switch overhead is not negligible?
  - Tick scheduling?
  - The deadline is earlier than the period?

Non-preemptable code section

- a non-preemptable code section (NPS) of a low priority task ***blocks*** high priority task
  - How to take this into account in time-demand analysis?

$$b_i = \max_{j=i+1}^{n} NPS_j$$

$$r_i = e_i + b_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i}{p_j} \right\rceil e_j$$

Non-preemptable code section

- a non-preemptable code section (NPS) of a low priority task ***blocks*** high priority task
  - How to take this into account in utilization bound check?

$$b_i = \max_{j=i+1}^{n} NPS_j$$

$$\sum_{j=1}^{i-1} \frac{e_j}{p_j} + \frac{e_i + b_i}{p_i} \le U(i);\ \text{task by task check}$$

$$\sum_{j=1}^{n} \frac{e_j}{p_j} + \max_{j=1}^{n} \frac{b_j}{p_j} \le U(n);\ \text{single check}$$

Deadline earlier than period?

- Time-demand analysis naturally works for this case
- What about utilization bound check?
  - Two utilization inflation methods

$$D_i < p_i$$

$$\sum_{j=1}^{i-1} \frac{e_j}{p_j} + \frac{e_i + p_i - D_i}{p_i} \le U(i);\ \text{increase execution time}$$

$$\sum_{j=1}^{i-1} \frac{e_j}{p_j} + \frac{e_i}{D_i} \le U(i);\ \text{decrease period}$$