Two approaches to just-in-time scheduling

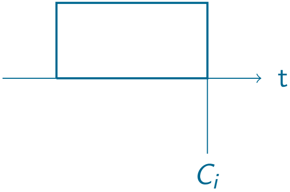
Joanna Józefowska

Poznan University of Technology

ICORES 2023
Lisbon, Portugal, February 21, 2023

Just-in-time scheduling

Scheduling objectives

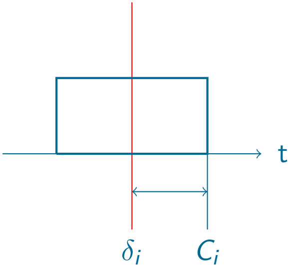


completion time: C_i

$$C_{max} = \max_i \{C_i\}$$

$$F = \sum_i (w_i) C_i$$

Scheduling objectives



completion time: C_i
due date (deadline): δ_i
lateness: $L_i = C_i - \delta_i$
(weighted) tardiness:
$T_i = \max\{0, C_i - \delta_i\}$

$$C_{max} = \max_i \{C_i\}$$

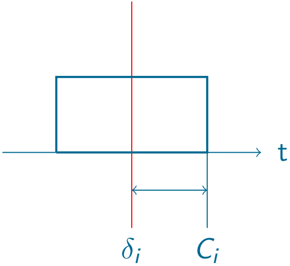
$$F = \sum_i (w_i) C_i$$

$$L_{max} = \max_i \{L_i\}$$

$$T = \sum_i (w_i) T_i$$

Earliness and tardiness costs

Assume the due date δ_i is given for each item i.

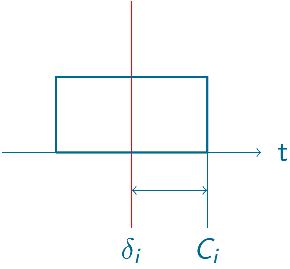


$$C_i - \delta_i > 0$$

Item i is completed late.

Earliness and tardiness costs

Assume the due date δ_i is given for each item i.



$$C_i - \delta_i > 0$$

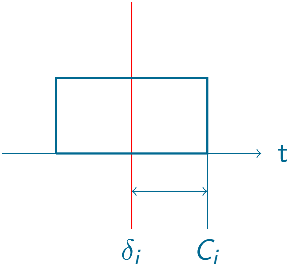
Item i is completed late.

"Tardiness" costs:

- financial penalties
- loss of order / client
- loss of reputation

Earliness and tardiness costs

Assume the due date δ_i is given for each item i.



$$C_i - \delta_i > 0$$

Item i is completed late.

"Tardiness" costs:

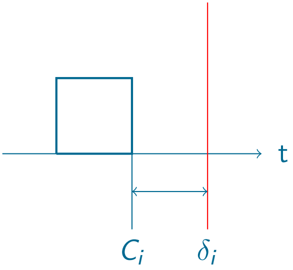
- financial penalties
- loss of order / client
- loss of reputation

Non-decreasing function $f_t(t_i)$ where

$$t_i = \begin{cases} C_i - \delta_i & \text{if } C_i > \delta_i \\ 0 & \text{otherwise} \end{cases}$$

Earliness and tardiness costs

Assume the due date δ_i is given for each item i.

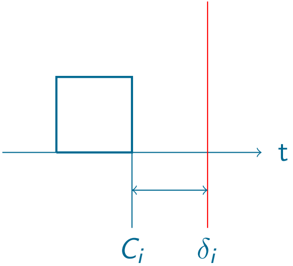


$$\delta_i - C_i > 0$$

Item i is completed "early".

Earliness and tardiness costs

Assume the due date δ_i is given for each item i.



$$\delta_i - C_i > 0$$

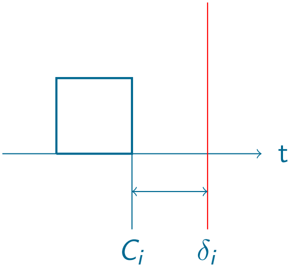
Item i is completed "early".

"Earliness" costs:

- storage space
- freezing of funds
- possible impairment of product

Earliness and tardiness costs

Assume the due date δ_i is given for each item i.



$$\delta_i - C_i > 0$$

Item i is completed "early".

"Earliness" costs:

- storage space
- freezing of funds
- possible impairment of product

Non-decreasing function $f_e(e_i)$ where

$$e_i = \begin{cases} \delta_i - C_i & \text{if } \delta_i > C_i \\ 0 & \text{otherwise} \end{cases}$$

Objective function

$$\text{minimize} \sum_{i=1}^{n} (f_e(e_i) + f_t(t_i))$$

Objective function

$$\text{minimize} \sum_{i=1}^{n} (f_e(e_i) + f_t(t_i))$$

Some special cases:

- linear cost functions $\sum_{i=1}^{n}(\alpha_i e_i + \beta_i t_i)$

Objective function

$$\text{minimize} \sum_{i=1}^{n} (f_e(e_i) + f_t(t_i))$$

Some special cases:

- linear cost functions $\sum_{i=1}^{n} (\alpha_i e_i + \beta_i t_i)$
  - total absolute deviation $\sum_{i=1}^{n} |C_i - \delta_i|$

Objective function

$$\text{minimize} \sum_{i=1}^{n} (f_e(e_i) + f_t(t_i))$$

Some special cases:

- linear cost functions $\sum_{i=1}^{n}(\alpha_i e_i + \beta_i t_i)$
  - total absolute deviation $\sum_{i=1}^{n} |C_i - \delta_i|$
  - scheduling around a common due date $\sum_{i=1}^{n} |C_i - \delta|$ (Kanet 1981, Hall, Kubiak and Sethi 1991, Hoogeveen, van de Velde 1991)

Scheduling with a common due date

minimize $\sum_{i=1}^{n} |\delta - C_i|$

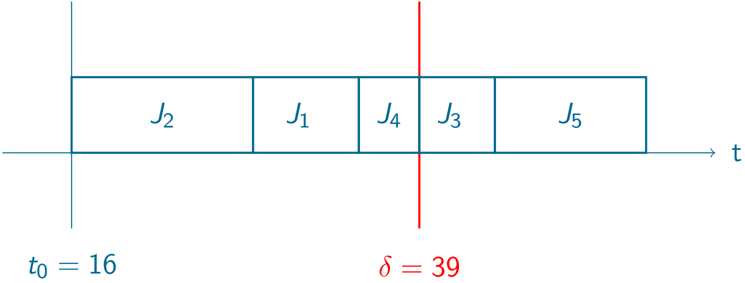
U - set of independent, nonpreemptable jobs, $|U| = n$
p_i - processing time of job i, $i = 1, \ldots, n$
δ - (large) common due date

Algorithm (Kanet, 1981)

```
B ← A ← ∅;
while (U ≠ 0) do
  remove a job k from U such that p_k = max_i{p_i};
  insert job k into the last position in B;
  if (U ≠ ∅) do
    remove a job k from U such that p_k = max_i{p_i};
    insert job k into the first position in A;
  end
end
S ← (B, A);
```

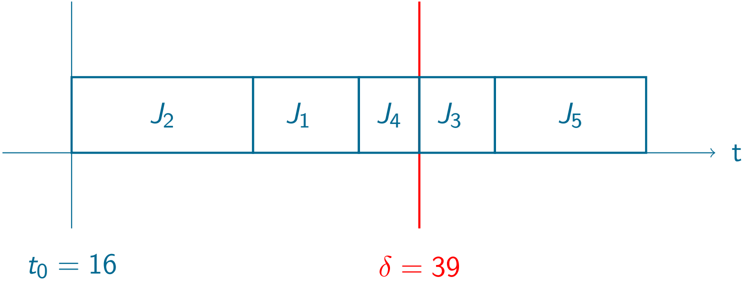
Example

i	1	2	3	4	5
p_i	7	12	5	4	10
$\delta = 39$					



Example

i	1	2	3	4	5
p_i	7	12	5	4	10
$\delta = 39$					



δ "restrictive" $(\delta < 23)$ – NP-hard

Properties of optimal schedules with common due date

- the longest job is scheduled first,

Properties of optimal schedules with common due date

- the longest job is scheduled first,
- V-shaped order of jobs,

- the longest job is scheduled first,
- V-shaped order of jobs,
- $|B| \geq |A|$,

Properties of optimal schedules with common due date

- the longest job is scheduled first,
- V-shaped order of jobs,
- $|B| \geq |A|$,
- $\sum_{i \in B} p_i \geq \sum_{i \in A} p_i$

Computational complexity

Problem	Complexity
$Qm\|\delta_i = \delta\|\sum(\alpha e_i + \beta t_i)$	$O(nlogn)$
$1\|\delta_i = \delta^{res}\|\sum\|C_i - \delta_i\|$	NP-hard
$1\|\|\sum\|C_i - \delta_i\|$	NP-hard

Objective function

$$\text{minimize} \sum_{i=1}^{n} (f_e(e_i) + f_t(t_i))$$

Some special cases:

- linear cost functions $\sum_{i=1}^{n}(\alpha_i e_i + \beta_i t_i)$
  - total absolute deviation $\sum_{i=1}^{n} |C_i - \delta_i|$
  - scheduling around a common due date $\sum_{i=1}^{n} |C_i - \delta|$ (Kanet 1981, Hall, Kubiak and Sethi 1991, Hoogeveen, van de Velde 1991)

Objective function

$$\text{minimize} \sum_{i=1}^{n} (f_e(e_i) + f_t(t_i))$$

Some special cases:

- linear cost functions $\sum_{i=1}^{n}(\alpha_i e_i + \beta_i t_i)$
  - total absolute deviation $\sum_{i=1}^{n}|C_i - \delta_i|$
  - scheduling around a common due date $\sum_{i=1}^{n}|C_i - \delta|$ (Kanet 1981, Hall, Kubiak and Sethi 1991, Hoogeveen, van de Velde 1991)
- non-linear cost functions $\sum_{i=1}^{n}|C_i - \delta|^{\alpha}, \alpha \in \mathbb{R}$

Objective function

$$\text{minimize} \sum_{i=1}^{n} (f_e(e_i) + f_t(t_i))$$

Some special cases:

- linear cost functions $\sum_{i=1}^{n}(\alpha_i e_i + \beta_i t_i)$
  - total absolute deviation $\sum_{i=1}^{n} |C_i - \delta_i|$
  - scheduling around a common due date $\sum_{i=1}^{n} |C_i - \delta|$ (Kanet 1981, Hall, Kubiak and Sethi 1991, Hoogeveen, van de Velde 1991)
- non-linear cost functions $\sum_{i=1}^{n} |C_i - \delta|^{\alpha}, \alpha \in \mathbb{R}$
  - $\alpha \geq 2$ NP-hard (Kubiak 1993)

Objective function

$$\text{minimize} \sum_{i=1}^{n} (f_e(e_i) + f_t(t_i))$$

Some special cases:

- linear cost functions $\sum_{i=1}^{n}(\alpha_i e_i + \beta_i t_i)$
  - total absolute deviation $\sum_{i=1}^{n} |C_i - \delta_i|$
  - scheduling around a common due date $\sum_{i=1}^{n} |C_i - \delta|$ (Kanet 1981, Hall, Kubiak and Sethi 1991, Hoogeveen, van de Velde 1991)
- non-linear cost functions $\sum_{i=1}^{n} |C_i - \delta|^{\alpha}, \alpha \in \mathbb{R}$
  - $\alpha \geq 2$ NP-hard (Kubiak 1993)
  - $\alpha \leq 1$ polynomially solvable (for all concave functions)

Objective function

$$\text{minimize} \sum_{i=1}^{n} (f_e(e_i) + f_t(t_i))$$

Some special cases:

- linear cost functions $\sum_{i=1}^{n}(\alpha_i e_i + \beta_i t_i)$
  - total absolute deviation $\sum_{i=1}^{n} |C_i - \delta_i|$
  - scheduling around a common due date $\sum_{i=1}^{n} |C_i - \delta|$ (Kanet 1981, Hall, Kubiak and Sethi 1991, Hoogeveen, van de Velde 1991)
- non-linear cost functions $\sum_{i=1}^{n} |C_i - \delta|^\alpha, \alpha \in \mathbb{R}$
  - $\alpha \geq 2$ NP-hard (Kubiak 1993)
  - $\alpha \leq 1$ polynomially solvable (for all concave functions)
  - $1 < \alpha < 2$ open

Mass production and personalized orders

10%

15%

50%

25%

Mass production and personalized orders

 10%

 15%

 50%

 25%

Let us assume that assembly of each variant takes the same amount of time (1 unit) and that we need a schedule for 20 time units.

Mass production and personalized orders

	10%	2 pcs.
	15%	3 pcs.
	25%	5 pcs.
	50%	10 pcs.

1 (green)	2 (green)	3 (red)	4 (red)	5 (red)	6 (purple)	7 (purple)	8 (purple)	9 (purple)	10 (purple)	11 (yellow)	12 (yellow)	13 (yellow)	14 (yellow)	15 (yellow)	16 (yellow)	17 (yellow)	18 (yellow)	19 (yellow)	20 (yellow)

Mass production and personalized orders

 10% 2 pcs.

 15% 3 pcs.

 25% 5 pcs.

 50% 10 pcs.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Goal: The number of items completed up to time t as close as possible to amount proportional to product rate.

Mass production and personalized orders

	10%	2 pcs.	$\frac{20}{2} = 10$
	15%	3 pcs.	$\frac{20}{3} = 6\frac{2}{3}$
	25%	5 pcs.	$\frac{20}{4} = 5$
	50%	10 pcs.	$\frac{20}{10} = 2$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Mass production and personalized orders

	10%	2 pcs.	$\frac{20}{2} = 10$
	15%	3 pcs.	$\frac{20}{3} = 6\frac{2}{3}$
	25%	5 pcs.	$\frac{20}{4} = 5$
	50%	10 pcs.	$\frac{20}{10} = 2$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Notation

D	total demand
n	number of product variants
d_i	demand of variant $i, i = 1, \dots, n$
$r_i = \frac{d_i}{D}$	product rate of variant $i, i = 1, \dots, n$
x_{it}	number of items of variant $i, i = 1, \dots, n$ completed up to time $t, t = 1, \dots, D$

Notation

D	total demand
n	number of product variants
d_i	demand of variant $i, i = 1, \ldots, n$
$r_i = \frac{d_i}{D}$	product rate of variant $i, i = 1, \ldots, n$
x_{it}	number of items of variant $i, i = 1, \ldots, n$ completed up to time $t, t = 1, \ldots, D$

- Ideal number of copies of variant i completed up to time t equals tr_i.

Notation

D	total demand
n	number of product variants
d_i	demand of variant $i, i = 1, \ldots, n$
$r_i = \frac{d_i}{D}$	product rate of variant $i, i = 1, \ldots, n$
x_{it}	number of items of variant $i, i = 1, \ldots, n$ completed up to time $t, t = 1, \ldots, D$

- Ideal number of copies of variant i completed up to time t equals tr_i.
- The goal is to minimize the deviation from this ideal.

Mathematical model

$$\text{minimize} \sum_{i=1}^{n}\sum_{t=1}^{D}|x_{it} - tr_i|$$

$$\text{minimize} \sum_{i=1}^{n}\sum_{t=1}^{D}(x_{it} - tr_i)^2$$

$$\text{minimize} \max_{1\le i\le n}\max_{1\le t\le D}|x_{it} - tr_i|$$

subject to:

$$\sum_{i=1}^{n} x_{it} = t, t = 1, \ldots, D$$

$$0 \le x_{it+1} - x_{it} \le 1, i = 1, \ldots, n; t = 1, \ldots, D$$

$$x_{iD} = d_i, i = 1, \ldots, n$$

$$x_{it} \in \mathcal{N}^+ \cup \{0\}, i = 1, \ldots, n; t = 1, \ldots, D$$

Mathematical model

$$\text{minimize} \sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$$

$$\text{minimize} \sum_{i=1}^{n} \sum_{t=1}^{D} (x_{it} - tr_i)^2$$

$$\text{minimize} \max_{1 \le i \le n} \max_{1 \le t \le D} |x_{it} - tr_i|$$

subject to:

$$\sum_{i=1}^{n} x_{it} = t, t = 1, \ldots, D$$

$$0 \le x_{it+1} - x_{it} \le 1, i = 1, \ldots, n; t = 1, \ldots, D$$

$$x_{iD} = d_i, i = 1, \ldots, n$$

$$x_{it} \in \mathcal{N}^+ \cup \{0\}, i = 1, \ldots, n; t = 1, \ldots, D$$

Mathematical model

$$\text{minimize} \sum_{i=1}^{n}\sum_{t=1}^{D}|x_{it} - tr_i|$$

$$\text{minimize} \sum_{i=1}^{n}\sum_{t=1}^{D}(x_{it} - tr_i)^2$$

$$\text{minimize} \max_{1\leq i\leq n}\max_{1\leq t\leq D}|x_{it} - tr_i|$$

subject to:

$$\sum_{i=1}^{n} x_{it} = t, t = 1, \ldots, D$$

$$0 \leq x_{it+1} - x_{it} \leq 1, i = 1, \ldots, n; t = 1, \ldots, D$$

$$x_{iD} = d_i, i = 1, \ldots, n$$

$$x_{it} \in \mathcal{N}^+ \cup \{0\}, i = 1, \ldots, n; t = 1, \ldots, D$$

Mathematical model

$$\text{minimize} \sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$$

$$\text{minimize} \sum_{i=1}^{n} \sum_{t=1}^{D} (x_{it} - tr_i)^2$$

$$\text{minimize} \max_{1 \le i \le n} \max_{1 \le t \le D} |x_{it} - tr_i|$$

subject to:

$$\sum_{i=1}^{n} x_{it} = t, t = 1, \ldots, D$$

$$0 \le x_{it+1} - x_{it} \le 1, i = 1, \ldots, n; t = 1, \ldots, D$$

$$x_{iD} = d_i, i = 1, \ldots, n$$

$$x_{it} \in \mathcal{N}^+ \cup \{0\}, i = 1, \ldots, n; t = 1, \ldots, D$$

Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

Z_j - completion time of the j-th copy of a product

$$\sum_{t=1}^{D} |x_t - tr|$$

Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

Z_j - completion time of the j-th copy of a product

$$\sum_{t=1}^{D} |x_t - tr| = \sum_{t=1}^{Z_1 - 1} |0 - tr| +$$

Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

Z_j - completion time of the j-th copy of a product

$$\sum_{t=1}^{D} |x_t - tr| = \sum_{t=1}^{Z_1 - 1} |0 - tr| + \sum_{t=Z_1}^{Z_2 - 1} |1 - tr| + \ldots + \sum_{t=Z_d}^{D} |d - tr|$$

Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

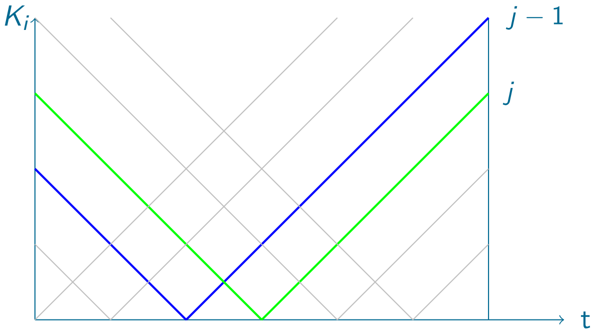
Z_j - completion time of the j-th copy of a product

$$\sum_{t=1}^{D} |x_t - tr| = \sum_{t=1}^{Z_1-1} |0 - tr| + \sum_{t=Z_1}^{Z_2-1} |1 - tr| + \ldots + \sum_{t=Z_d}^{D} |d - tr|$$

Z_j^* - ideal completion time of the j-th copy of a product.

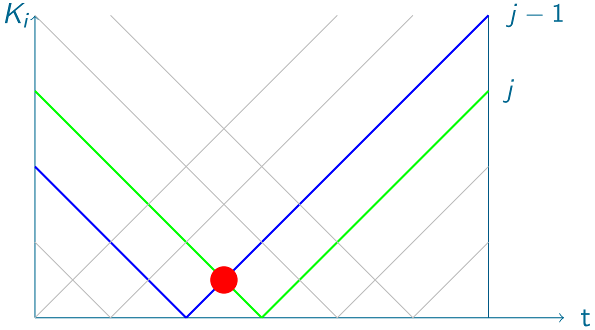
Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

Z_j^{i*} - ideal completion time of the j-th copy of product i.



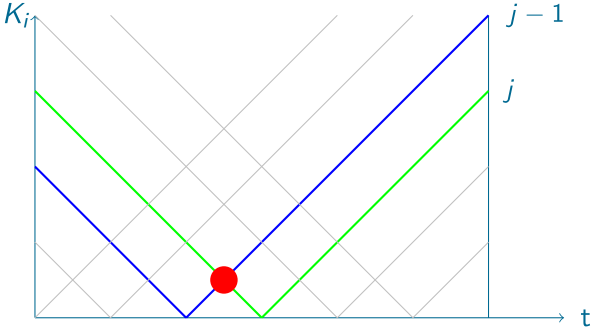
Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

Z_j^{i*} - ideal completion time of the j-th copy of product i.



Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

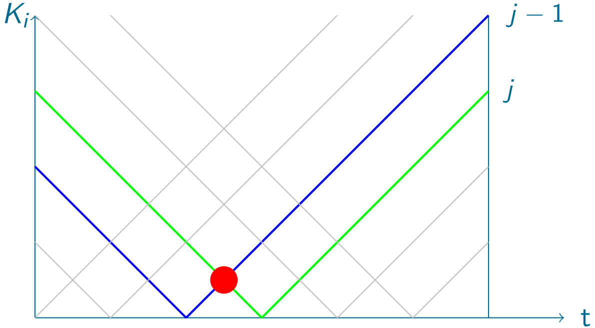
Z_j^{i*} - ideal completion time of the j-th copy of product i.



$tr_i - (j-1) = -(tr_i - j)$

Kubiak&Sethi: minimize $\sum_{i=1}^{n}\sum_{t=1}^{D}|x_{it} - tr_i|$

Z_j^{i*} - ideal completion time of the j-th copy of product i.

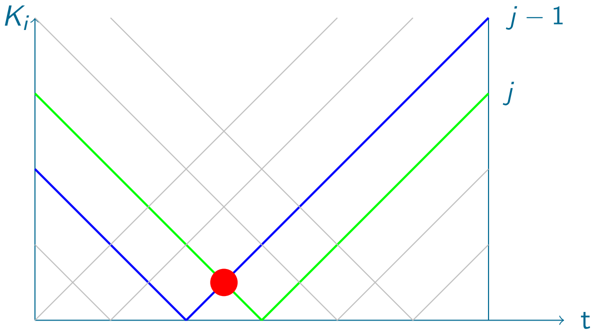


$tr_i - (j-1) = -(tr_i - j)$

$t = \frac{2j-1}{2r_i}$

Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

Z_j^{i*} - ideal completion time of the j-th copy of product i.



$$Z_j^{i*} = \left\lceil \frac{2j-1}{2r_i} \right\rceil$$

Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

1. calculate ideal position of (i, j):

$$Z_j^{i*} = \left\lceil \frac{2j - 1}{2r_i} \right\rceil$$

2. calculate the cost C_{jt}^i of scheduling (i, j) in position t:

$$C_{jt}^i = \sum_{l=min(t, Z_j^{i*})}^{max(t, Z_j^{i*})-1} ||lr_i - j| - |lr_i - (j-1)||$$

Kubiak&Sethi: minimize $\sum_{i=1}^{n} \sum_{t=1}^{D} |x_{it} - tr_i|$

Optimal solution is found by solving the following assignment problem:

$$\text{minimize} \sum_{(i,j)\in J} \sum_{t=1}^{D} C^i_{jt} y^i_{jt}$$

$$\text{s.t.} \quad \sum_{t=1}^{D} y^i_{jt} = 1$$

$$\sum_{(i,j)\in J} y^i_{jt} = 1$$

$$(i,j) \in J \Leftrightarrow i \in \{1,2,\ldots,n\} \vee j \in \{1,2,\ldots,d_i\}$$

$$y^i_{jt} = \begin{cases} 1 & \text{if } j\text{-th copy of } i \text{ completes in } t \\ 0 & \text{otherwise} \end{cases}$$

Steiner&Yeomans: minimize $\max_{it} |x_{it} - tr_i|$

Theorem

A just in time sequence with

$$\max_{it} |x_{it} - tr_i| \leq T$$

exists if and only if there exists a sequence that allocates the j-th copy of product i in the interval $[E(i,j), L(i,j)]$ where

$$E(i,j) = \left\lceil \frac{1}{r_i}(j - T) \right\rceil \qquad L(i,j) = \left\lfloor \frac{1}{r_i}(j - 1 + T) + 1 \right\rfloor$$

Steiner&Yeomans: minimize $\max_{it} |x_{it} - tr_i|$

Theorem

A just in time sequence with

$$\max_{it} |x_{it} - tr_i| \leq T \leq 1 - \frac{1}{D}$$

exists if and only if there exists a sequence that allocates the j-th copy of product i in the interval $[E(i,j), L(i,j)]$ where

$$E(i,j) = \left\lceil \frac{1}{r_i}(j - T) \right\rceil \qquad L(i,j) = \left\lfloor \frac{1}{r_i}(j - 1 + T) + 1 \right\rfloor$$

The algorithm tests values $T \in \left\{ \frac{D-d_{max}}{D}, \frac{D-d_{max+1}}{D}, \ldots, \frac{D-1}{D} \right\}$ in ascending order.

Problem of apportionment

Given are:

- the number of states s,
- an integer vector of populations: $\mathbf{p} = (p_1, p_2, p_3, \ldots, p_s)$
- an integer size of the house, $h \geq 0$

Problem of apportionment

Given are:

- the number of states s,
- an integer vector of populations: $\mathbf{p} = (p_1, p_2, p_3, \ldots, p_s)$
- an integer size of the house, $h \geq 0$

An apportionment of h seats among s states is an integer vector $\mathbf{a}$ such that:

$$\mathbf{a} = (a_1, a_2, a_3, ..., a_s)$$

$$\sum_{i=1}^{s} a_i = h.$$

Goal: Find a fair apportionment.

Problem of apportionment

Given are:

- the number of states s,
- an integer vector of populations: $\mathbf{p} = (p_1, p_2, p_3, \ldots, p_s)$
- an integer size of the house, $h \geq 0$

An apportionment of h seats among s states is an integer vector $\mathbf{a}$ such that:

$$\mathbf{a} = (a_1, a_2, a_3, \ldots, a_s)$$

$$\sum_{i=1}^{s} a_i = h.$$

Goal: Find a fair apportionment.

Objective functions - criteria

Hamilton: $\sum_{i=1}^{s} \left| \frac{a_i}{h} - \frac{p_i}{\sum_i p_i} \right|$

Objective functions - criteria

Hamilton:	$\sum_{i=1}^{s} \left\| \frac{a_i}{h} - \frac{p_i}{\sum_i p_i} \right\|$
Webster, Hamilton:	$\sum_{i=1}^{s} f\left(a_i - \frac{hp_i}{\sum_i p_i}\right)$ where f is any l_p norm
Hill:	$\sum_{i=1}^{s} a_i \left(\frac{p_i}{a_i} - \frac{\sum_i p_i}{h}\right)^2$
Burt & Harris:	$\max_{i,j} \left\{ \frac{p_i}{a_i} - \frac{p_j}{a_j} \right\}$
	$\max_{i,j} \left\{ \frac{a_i}{p_i} - \frac{a_j}{p_j} \right\}$
Jefferson:	$\max_i \left\{ \frac{a_i}{p_i} \right\}$
Adams:	$\max_i \left\{ \frac{p_i}{a_i} \right\}$

Alabama paradox

It was observed in 1880 by C. W. Seaton (chief clerk of U. S. Census Office, USA) that in a house of 299 seats Alabama receives 8 seats while in a house of 300 seats Alabama receives 7 seats.

Alabama paradox

It was observed in 1880 by C. W. Seaton (chief clerk of U. S. Census Office, USA) that in a house of 299 seats Alabama receives 8 seats while in a house of 300 seats Alabama receives 7 seats.

Hamilton method

1. Allocate $\left\lfloor \frac{p_i}{SD} \right\rfloor$ seats to each state $i, i = 1, \ldots, s$, where

$$SD = \frac{\sum_{i=1}^{s} p_i}{h}$$

Alabama paradox

It was observed in 1880 by C. W. Seaton (chief clerk of U. S. Census Office, USA) that in a house of 299 seats Alabama receives 8 seats while in a house of 300 seats Alabama receives 7 seats.

Hamilton method

1. Allocate $\left\lfloor \frac{p_i}{SD} \right\rfloor$ seats to each state $i, i = 1, \ldots, s$, where

$$SD = \frac{\sum_{i=1}^{s} p_i}{h}$$

2. Assign the remaining seats to the states with biggest fractional value of $\frac{p_i}{SD}$.

Alabama paradox - example

Hamilton method

1. Allocate $\left\lfloor \frac{p_i}{SD} \right\rfloor$ seats to each state $i, i = 1, \ldots, s$, where

$$SD = \frac{\sum_{i=1}^{s} p_i}{h}$$

2. Assign the remaining seats to the states with biggest fractional value of $\frac{p_i}{SD}$).

state		$h = 21$			$h = 22$		
i	p_i	$\frac{p_i}{SD}$	$\left\lfloor \frac{p_i}{SD} \right\rfloor$	a_i	$\frac{p_i}{SD}$	$\left\lfloor \frac{p_i}{SD} \right\rfloor$	a_i
A	7 270	14.24	14	14	14.92	14	15
B	1 230	2.41	2	3	2.52	2	2
C	2 220	4.35	4	4	4.56	4	5
Total	10 720	22.00	20	21	22.00	20	22

Alabama paradox - example

Hamilton method

1. Allocate $\left\lfloor \frac{p_i}{SD} \right\rfloor$ seats to each state $i, i = 1, \ldots, s$, where

$$SD = \frac{\sum_{i=1}^{s} p_i}{h}$$

2. Assign the remaining seats to the states with biggest fractional value of $\frac{p_i}{SD}$).

state		$h = 21$			$h = 22$		
i	p_i	$\frac{p_i}{SD}$	$\left\lfloor \frac{p_i}{SD} \right\rfloor$	a_i	$\frac{p_i}{SD}$	$\left\lfloor \frac{p_i}{SD} \right\rfloor$	a_i
A	7 270	14.24	14	14	14.92	14	15
B	1 230	2.41	2	3	2.52	2	2
C	2 220	4.35	4	4	4.56	4	5
Total	10 720	22.00	20	21	22.00	20	22

Alabama paradox - example

Hamilton method

1. Allocate $\left\lfloor \frac{p_i}{SD} \right\rfloor$ seats to each state $i, i = 1, \ldots, s$, where

$$SD = \frac{\sum_{i=1}^{s} p_i}{h}$$

2. Assign the remaining seats to the states with biggest fractional value of $\frac{p_i}{SD}$).

state		$h = 21$			$h = 22$		
i	p_i	$\frac{p_i}{SD}$	$\left\lfloor \frac{p_i}{SD} \right\rfloor$	a_i	$\frac{p_i}{SD}$	$\left\lfloor \frac{p_i}{SD} \right\rfloor$	a_i
A	7 270	14.24	14	14	14.92	14	15
B	1 230	2.41	2	3	2.52	2	2
C	2 220	4.35	4	4	4.56	4	5
Total	10 720	22.00	20	21	22.00	20	22

House monotonicity

Balinski & Young, 1970

House monotone methods

An apportionment method is called house monotone if the number of seats assigned to any state $i, i = 1, \dots, n$, in a parliament of size $h + 1$ is greater than or equal to the number of seats assigned to the same state in a house of size h.

House monotonicity

Balinski & Young, 1970

House monotone methods

An apportionment method is called house monotone if the number of seats assigned to any state $i, i = 1, \ldots, n$, in a parliament of size $h + 1$ is greater than or equal to the number of seats assigned to the same state in a house of size h.

Property

Hamilton method is not house monotone.

Population monotonicity

Population monotone methods

An apportionment method M is called population monotone if for any two vectors of populations $p, p^{'} > 0$ and vectors of apportionments $a \in (M, h), a' \in (M, h)$ the following implication holds:

$$\frac{p'_{i'}}{p'_{j'}} \geq \frac{p_i}{p_j} \Rightarrow \begin{cases} a'_{i'} \geq a_i \text{ or } a'_{j'} \leq a_j \\ \text{or} \\ \frac{p'_{i'}}{p'_{j'}} = \frac{p_i}{p_j} \text{ and } a'_{i'}, a'_{j'} \text{ can be substituted for } a_i, a_j \text{ in } \mathbf{a} \end{cases}$$

Population monotonicity

Population monotone methods

An apportionment method M is called population monotone if for any two vectors of populations $p, p^{'} > 0$ and vectors of apportionments $a \in (M, h), a' \in (M, h)$ the following implication holds:

$$\frac{p'_{i'}}{p'_{j'}} \geq \frac{p_i}{p_j} \Rightarrow \begin{cases} a'_{i'} \geq a_i \text{ or } a'_{j'} \leq a_j \\ \text{or} \\ \frac{p'_{i'}}{p'_{j'}} = \frac{p_i}{p_j} \text{ and } a'_{i'}, a'_{j'} \text{ can be substituted for } a_i, a_j \text{ in } \mathbf{a} \end{cases}$$

Property (Balinski & Young)

Any population monotone method is house monotone but not vice versa.

Divisor methods

Divisor methods

Assign the next seat to the state with maximum value of $\frac{p_i}{d(a_i)}$, where $d(a_i)$ is divisor defined below.

Method	Divisor $d(a)$
Adams	a
Dean	$\frac{a(a+1)}{a+0.5}$
Hill	$\sqrt{a(a+1)}$
Webster	$a+0.5$
Jefferson	$a+1$

Divisor methods

Divisor methods

Assign the next seat to the state with maximum value of $\frac{p_i}{d(a_i)}$, where $d(a_i)$ is divisor defined below.

Method	Divisor $d(a)$
Adams	a
Dean	$\frac{a(a+1)}{a+0.5}$
Hill	$\sqrt{a(a+1)}$
Webster	$a+0.5$
Jefferson	$a+1$

Property (Balinski & Young)

An apportionment method is a divisor method iff it is population monotone.

Quota methods of apportionment

Quota methods

A method of apportionment stays within the quota if all its allocations satisfy the following inequalities:

$$\left\lfloor \frac{h \cdot p_i}{\sum_{i=1}^{s} p_i} \right\rfloor \leq a_i \leq \left\lceil \frac{h \cdot p_i}{\sum_{i=1}^{s} p_i} \right\rceil$$

Quota methods of apportionment

Quota methods

A method of apportionment stays within the quota if all its allocations satisfy the following inequalities:

$$\left\lfloor \frac{h \cdot p_i}{\sum_{i=1}^{s} p_i} \right\rfloor \leq a_i \leq \left\lceil \frac{h \cdot p_i}{\sum_{i=1}^{s} p_i} \right\rceil$$

Theorem (Balinski & Young)

No method of apportionment exists for $n \geq 4$ and $h \geq n + 3$ that is population monotone and stays within the quota.

Transformation

Scheduling	Apportionment
product $i, i = 1, \ldots, n$	state $i, i = 1, \ldots, s$
d_i demand for product i	p_i population of state i
time unit considered t	size of the house h
x_{it} cumulative number of copies of product i completed up to time t	a_i number of seats assigned to state i in a house of size h

Classification of methods

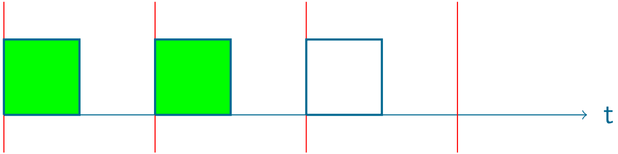
	HM	NHM
QM	Still, Steiner&Yeomans, quota-divisor methods	Hamilton
NQM	Kubiak&Sethi, divisor methods	

Real time systems (Liu-Layland, 1973)

- n periodic, preemptive and independent tasks
- single processor
- task $i, i = 1, \ldots, n$, is characterized by
  - its request period T_i
  - run-time C_i, such that $T_i \geq C_i$
- execution of the k-th request of task i, which occurs at time $(k-1)T_i$, must finish by the time kT_i

Real time systems (Liu-Layland, 1973)

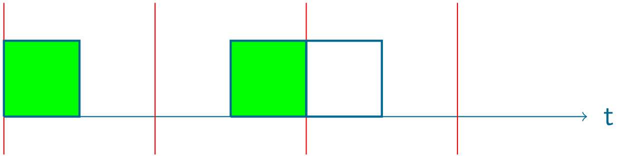
- n periodic, preemptive and independent tasks
- single processor
- task $i, i = 1, \ldots, n$, is characterized by
  - its request period T_i
  - run-time C_i, such that $T_i \geq C_i$
- execution of the k-th request of task i, which occurs at time $(k-1)T_i$, must finish by the time kT_i



Example: $T = 2, C = 1$

Real time systems (Liu-Layland, 1973)

- n periodic, preemptive and independent tasks
- single processor
- task $i, i = 1, \ldots, n$, is characterized by
  - its request period T_i
  - run-time C_i, such that $T_i \geq C_i$
- execution of the k-th request of task i, which occurs at time $(k-1)T_i$, must finish by the time kT_i

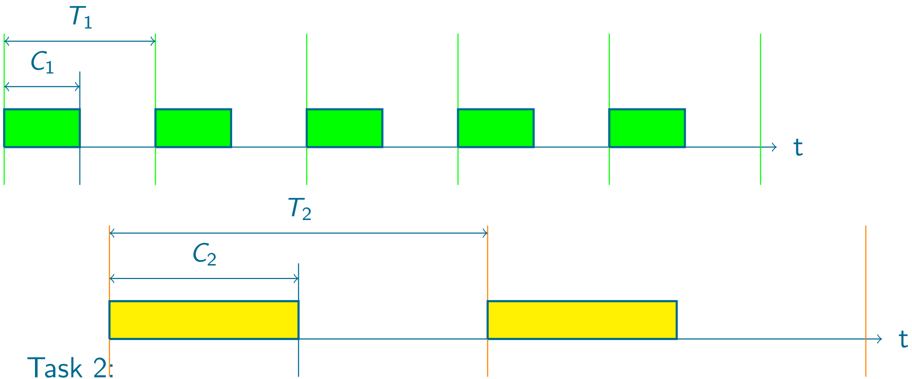


Example: $T = 2, C = 1$

Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4$, $C_1 = 2$, $T_2 = 10$, $C_2 = 5$

Task 1:

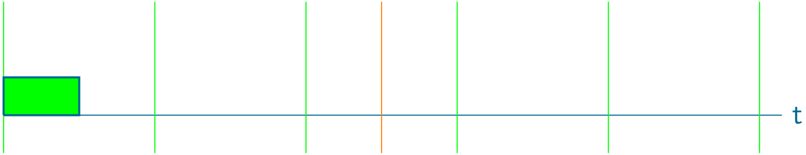


Task 2:

Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

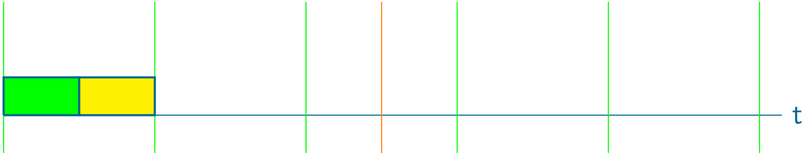
Rate monotonic algorithm: priority to tasks with smaller T_i.



Real time systems (Liu-Layland, 1973)

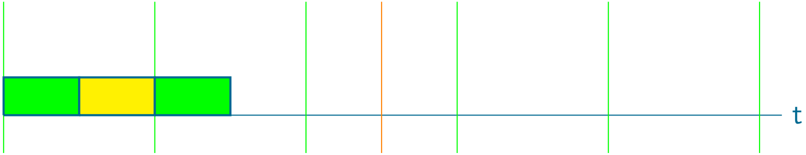
Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

Rate monotonic algorithm: priority to tasks with smaller T_i.



Real time systems (Liu-Layland, 1973)

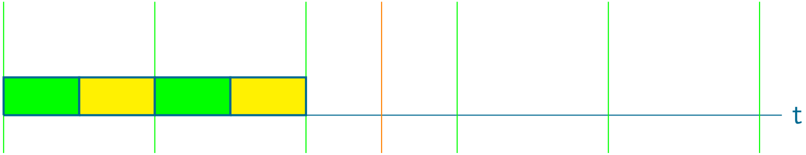
Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$
Rate monotonic algorithm: priority to tasks with smaller T_i.



Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

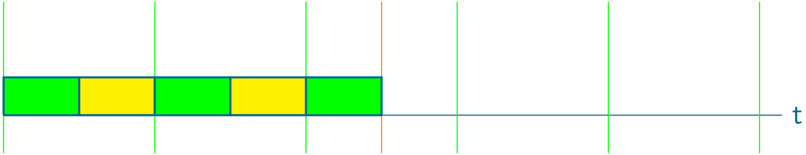
Rate monotonic algorithm: priority to tasks with smaller T_i.



Real time systems (Liu-Layland, 1973)

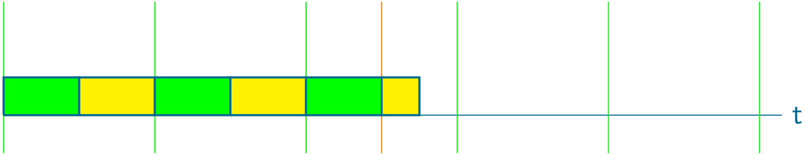
Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

Rate monotonic algorithm: priority to tasks with smaller T_i.



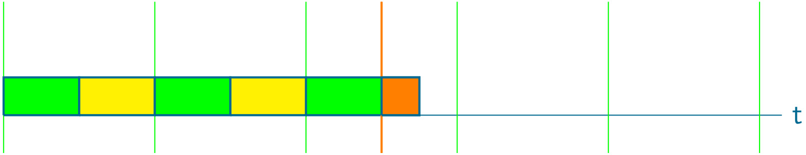
Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$
Rate monotonic algorithm: priority to tasks with smaller T_i.



Real time systems (Liu-Layland, 1973)

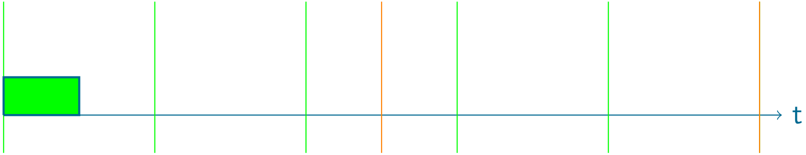
Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$
Rate monotonic algorithm: priority to tasks with smaller T_i.



Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

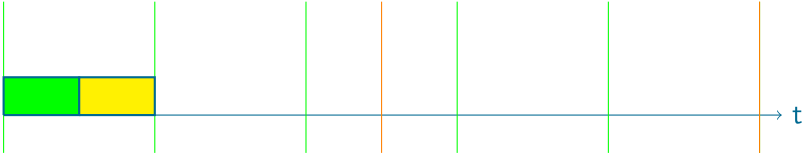
Deadline driven algorithm: priority to tasks with the closest deadline.



Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

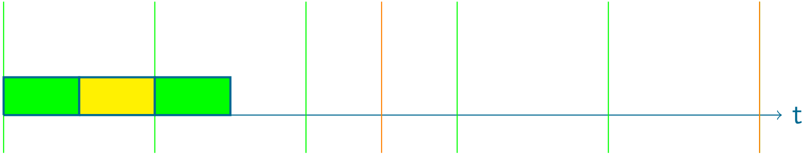
Deadline driven algorithm: priority to tasks with the closest deadline.



Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

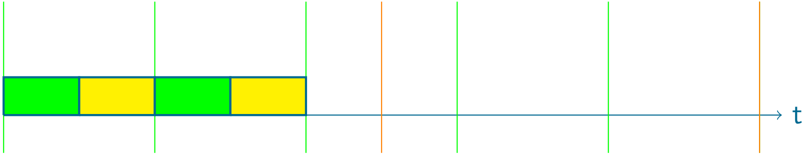
Deadline driven algorithm: priority to tasks with the closest deadline.



Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

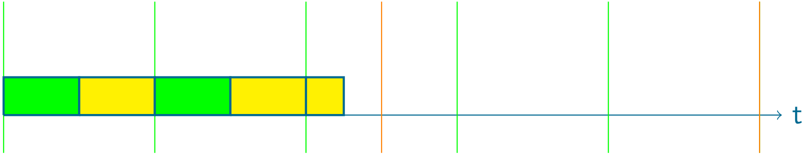
Deadline driven algorithm: priority to tasks with the closest deadline.



Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

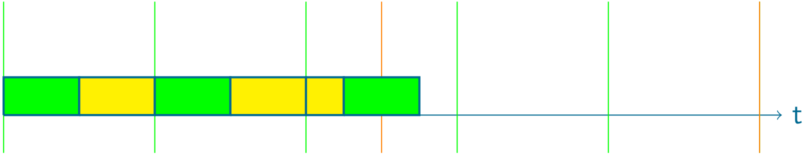
Deadline driven algorithm: priority to tasks with the closest deadline.



Real time systems (Liu-Layland, 1973)

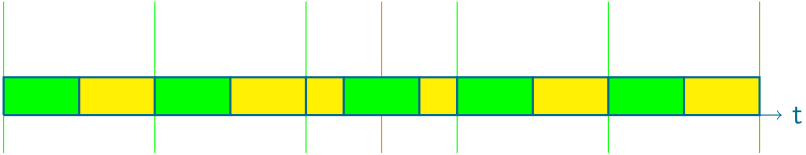
Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$

Deadline driven algorithm: priority to tasks with the closest deadline.



Real time systems (Liu-Layland, 1973)

Example: $T_1 = 4, C_1 = 2, T_2 = 10, C_2 = 5$
Deadline driven algorithm: priority to tasks with the closest deadline.



Liu-Layland problem and apportionment

- $\frac{C_i}{T_i}$ expresses the desired proportion of time units allocated to task i in a schedule of any given length - it corresponds to $\frac{p_i}{\sum p_i}$;
- the schedule length corresponds to the size of the house h;

Theorem (Kubiak 2004)

Any house monotone method satisfying the quota solves the Liu-Layland problem.

	HM	NHM
QM	Still, Steiner&Yeomans, quota-divisor methods	Hamilton
NQM	Kubiak&Sethi, divisor methods	

Liu-Layland problem and apportionment

	HM	NHM
QM	Still, Steiner&Yeomans, quota-divisor methods	Hamilton
NQM	Kubiak&Sethi, ~~divisor methods~~	

Theorem (Józefowska et.al. 2008)

Satisfying quota is a necessary condition for any divisor method to solve the Liu–Layland problem.

Corollary

No divisor method solves the Liu–Layland problem.

Stride scheduling

- n competing clients,
- w_i the importance of client i, $i = 1, 2, \ldots, n$,
- goal: allocating units of a discrete resource among clients in such a way that after any allocation the accumulated number of units of the resource possessed by client i is proportional to w_i,

Stride scheduling

- n competing clients,
- w_i the importance of client i, $i = 1, 2, ..., n$,
- goal: allocating units of a discrete resource among clients in such a way that after any allocation the accumulated number of units of the resource possessed by client i is proportional to w_i,
- dynamic environment: the number of clients n or the values w_i, $i = 1, 2, ..., n$, associated with clients may change in an unpredictable way.

Solution approach (Waldspurger and Weil, 1995)

- the problem of scheduling n processes on a single processor,

Solution approach (Waldspurger and Weil, 1995)

- the problem of scheduling n processes on a single processor,
- each client is assigned a number of tickets which are mapped to the values $w_i, i = 1, 2, \dots, n$,

Solution approach (Waldspurger and Weil, 1995)

- the problem of scheduling n processes on a single processor,
- each client is assigned a number of tickets which are mapped to the values $w_i, i = 1, 2, \dots, n$,
- the stride, inversely proportional to tickets, is calculated for each client,

Solution approach (Waldspurger and Weil, 1995)

- the problem of scheduling n processes on a single processor,
- each client is assigned a number of tickets which are mapped to the values $w_i, i = 1, 2, ..., n$,
- the stride, inversely proportional to tickets, is calculated for each client,
- pass represents the virtual time index for the clients next selection,

Solution approach (Waldspurger and Weil, 1995)

- the problem of scheduling n processes on a single processor,
- each client is assigned a number of tickets which are mapped to the values $w_i, i = 1, 2, ..., n$,
- the stride, inversely proportional to tickets, is calculated for each client,
- pass represents the virtual time index for the clients next selection,
- the client with minimum pass is selected and its pass is advanced by its stride,

Solution approach (Waldspurger and Weil, 1995)

- the problem of scheduling n processes on a single processor,
- each client is assigned a number of tickets which are mapped to the values $w_i, i = 1, 2, ..., n$,
- the stride, inversely proportional to tickets, is calculated for each client,
- pass represents the virtual time index for the clients next selection,
- the client with minimum pass is selected and its pass is advanced by its stride,
- after the quantum passes the process is preempted and the processor can be allocated to another client.

Summary

Stefan Banach (1892-1945)

A mathematician is a person who can find analogies between theorems;

Summary

Stefan Banach (1892-1945)

A mathematician is a person who can find analogies between theorems;
a better mathematician is one who can see analogies between proofs

Summary

Stefan Banach (1892-1945)

A mathematician is a person who can find analogies between theorems;
a better mathematician is one who can see analogies between proofs
and the best mathematician can notice analogies between theories.

Summary

Stefan Banach (1892-1945)

A mathematician is a person who can find analogies between theorems; a better mathematician is one who can see analogies between proofs and the best mathematician can notice analogies between theories. One can imagine that the ultimate mathematician is one who can see analogies between analogies.

Bibliography I

- Balinski ML, Young HF (1975) The quota method of apportionment. American Mathematical Monthly 82:701–730
- Gordon V, Proth J-M, Chu C (2002) Due date assignment and scheduling: SLK, TWK and other due date assignment models. Production Planning and Control 13:117–132
- Gordon V, Proth J-M, Chu Ch (2002) A survey of the state-of-the-art of common due date assignment and scheduling research. European Journal of Operational Research 139:1–25
- Hall NG, Kubiak W, Sethi SP (1991) Earliness–tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. Operations Research 39:847–856
- Hoogeveen JA, van de Velde SL (1991) Scheduling around a small common due date. European Journal of Operational Research 55:237–242
- Józefowska J (2007) Just-in-Time Scheduling. Models and Algorithms for Computer and Manufacturing Systems. Springer

Bibliography II

- Józefowska J, Józefowski Ł, Kubiak W, (2006) Characterization of just in time sequencing via apportionment. in: Yan H., Yin G., Zhang Q. (Eds.) Stochastic Processes, Optimization, and Control Theory Applications in Financial Engineering, Queuing Networks, and Manufacturing Systems/ A Volume in Honor of Suresh Sethi, Series: International Series in Operations Research & Management Science. Vol. 94, Springer Verlag 175–200
- Józefowska J, Józefowski Ł, Kubiak W. (2009) Apportionment methods and the Liu-Layland problem. European Journal of Operational Research 193: 857–864
- Kanet JJ (1981) Minimizing the average deviation of job completion times about a common due date. Naval Research Logistics Quarterly 28:643–651
- Kubiak W, (1993) Completion time variance minimization on a single machine is difficult. Oper. Res. Lett. 14: 49–59
- Kubiak W, Sethi S (1994) Optimal Just-in-Time Schedules for Flexible Transfer Lines. The International Journal of Flexible Manufacturing Systems 6:137–154
- Liu CL, Layland JW. Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment. (1973) Journal of ACM 20; 46–61

Bibliography III

- Mitlenburg J (1989) Level schedules for mixed-model assembly lines in just-in-time production systems. Management Science 35:192–207
- Steiner G, Yeomans S (1993) Level schedules for mixed-model Just-in-Time processes. Management Science 39:728–735
- Steiner G, Yeomans S (1996) Optimal level schedules in mixed-model multi-level JIT assembly systems with pegging. European Journal of Operational Research 95:38–52
- Still JW (1979) A class of new methods for Congressional Apportionment. SIAM Journal of Applied Mathematics 37:401–418