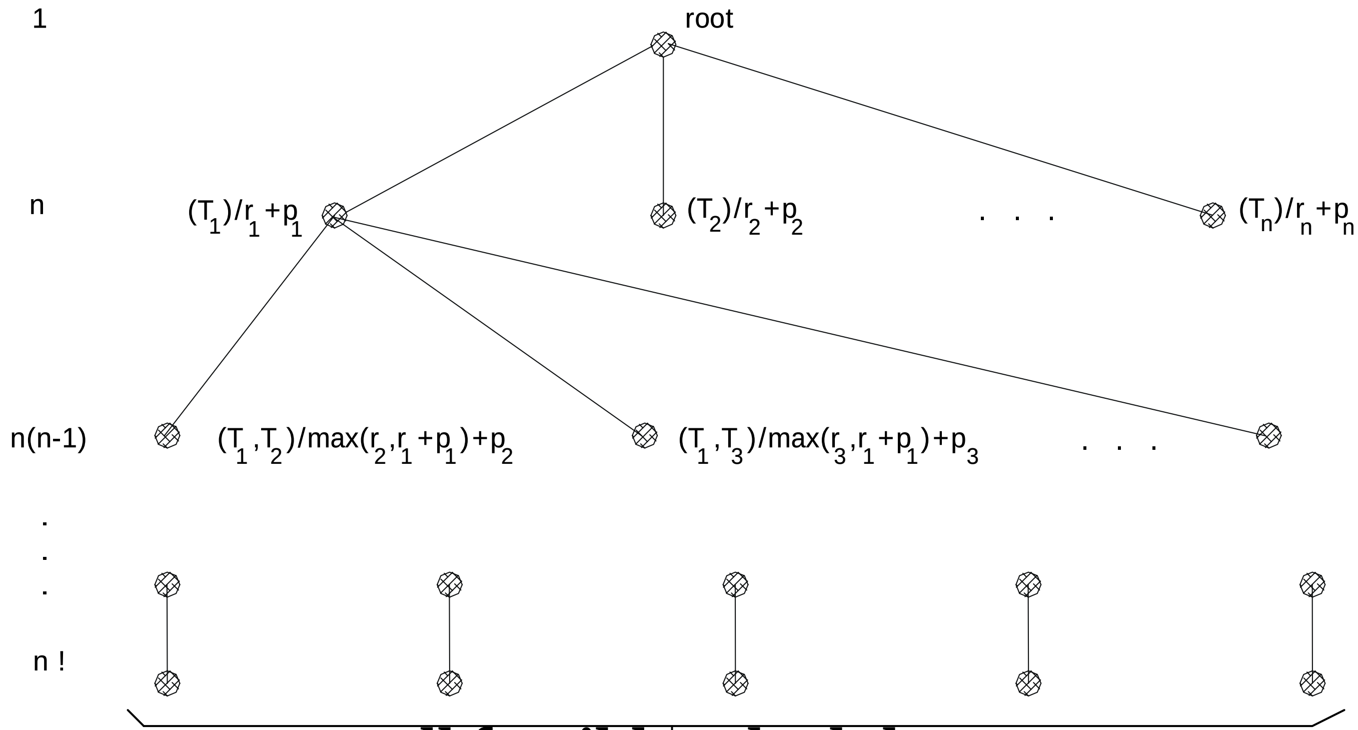
Single processor scheduling C_{max}

- $1|prec|C_{max}$ – simple
  - if tasks are assigned in whatever order in accordance with precedence relation, then $C_{max}=\Sigma p_j$
- $1||C_{max}$ – simple
- $1|r_j|C_{max}$ – simple
  - tasks are scheduled in order of nondecreasing release times
- $1|d_j^{\sim}|C_{max}$ – simple
  - tasks are scheduled in order of nondecreasing deadlines (**EDF** – earliest deadline first)
  - EDF provides optimal solution iff there exists a schedule that meets all the deadlines

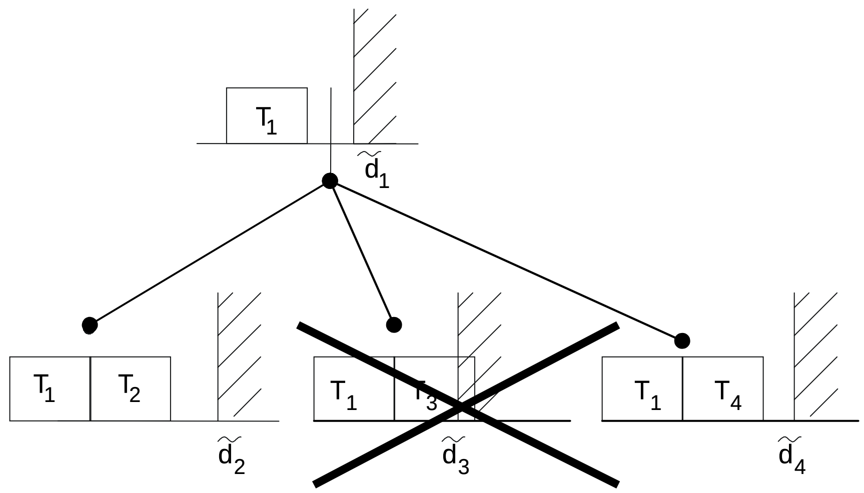
- $1|\ r_j, \tilde{d_j}\ |C_{max}$ – NP hard problem
    - transfromation from the 3-PARTITION problem
    - polynomial algorithm can be found if $p_j=1$
    - general problem can be solved by applying **Branch&Bound** algorithm by **Bratley**



all feasible schedules

(i) exceeding deadlines

– if completion time associated with at least one of the nodes under node *v* level *k-1* then all nodes under *v* can be eliminated

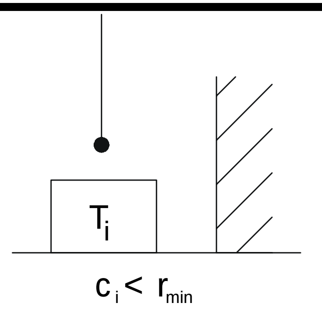


due to this vertex it is needed to eliminate both “brother” vertices

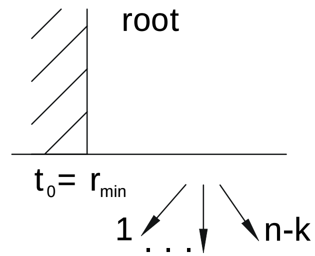
---

(ii) problem decomposition – if the completion time C_i of all scheduled tasks is less than or equal to smallest release time of all unscheduled tasks

situation at level k



it remains to schedule (n-k) tasks



Optimality test of Bratley's algorithm

block is a group of tasks such that the first task starts at its release time and all the following tasks to the end of the schedule are processed without idle time

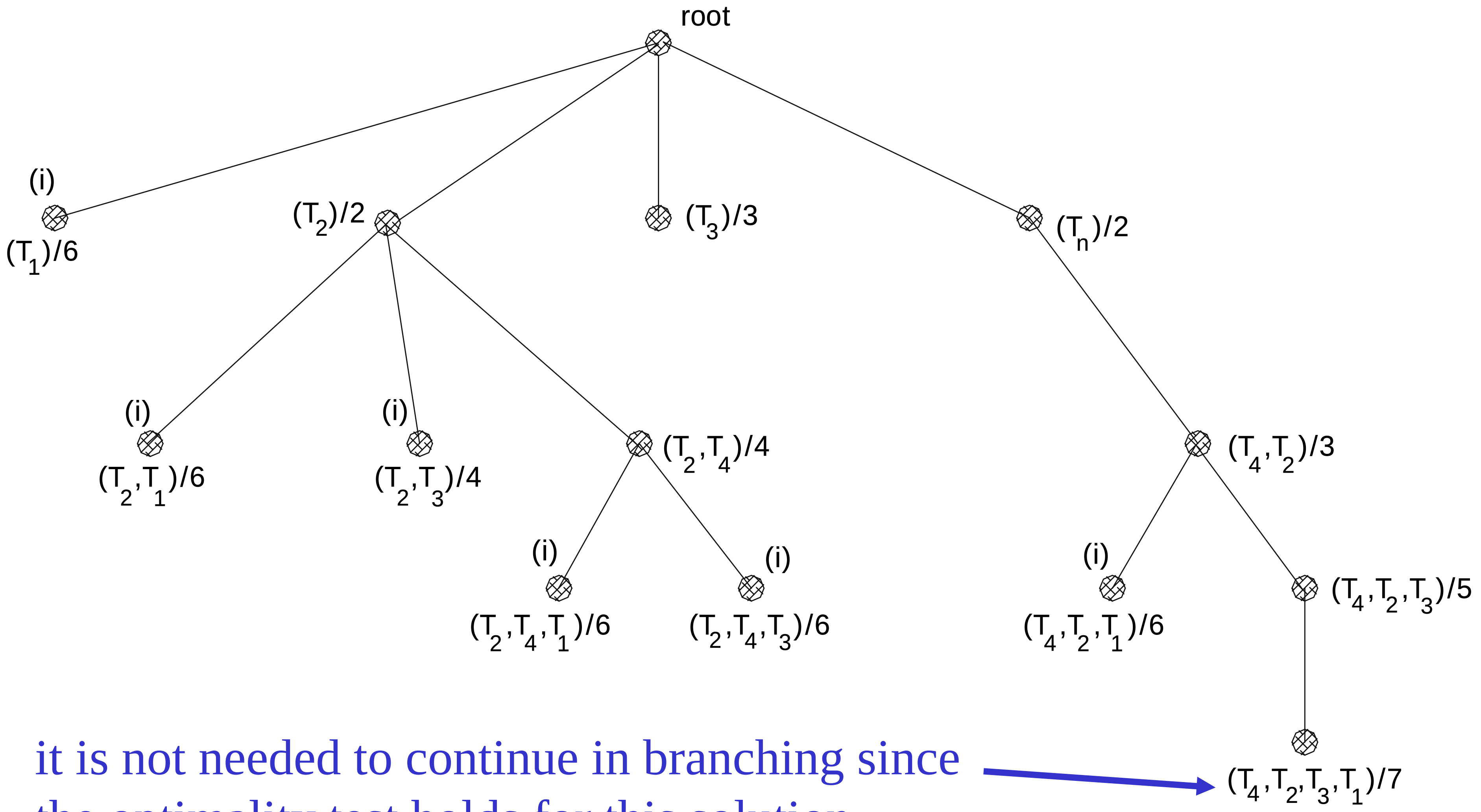
block satisfies **release time property** if release time of all tasks in the block are greater or equal to the release time of the fist task in the block

Lemma: A schedule is optimal iff it contains a block that satisfies the release time property

Proof:

- if part (each schedule with block satisfying RTP is optimal) - follows from the definition of RTP
- only if part (each optimal schedule has block satisfying RTP) – by contradiction – suppose schedules that do not have block with RTP, none of them is optimal

Example: r =[4,1,1,0], p =[2,1,2,2], d~ =[7,5,6,4]



it is not needed to continue in branching since
the optimality test holds for this solution

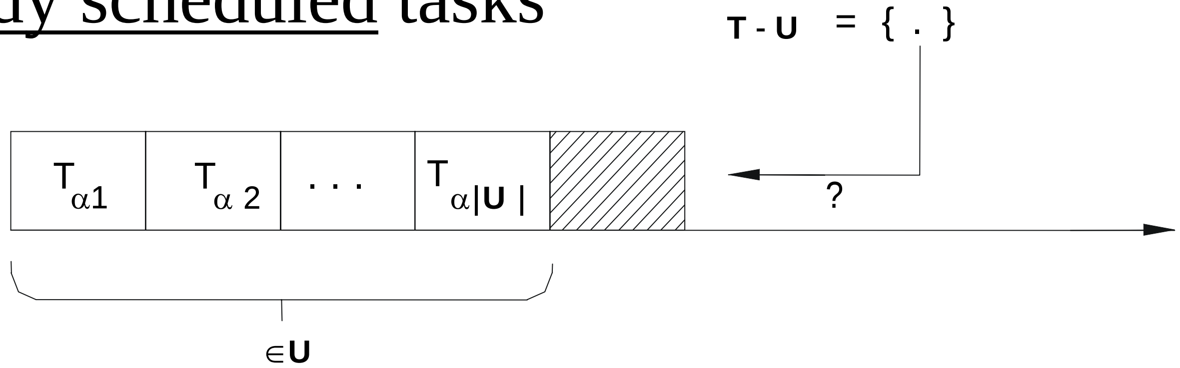
$\Sigma w_j C_j$

- $1||\Sigma C_j$ – Shortest Processing Time (SPT) rule
  - tasks are scheduled in order of nondecreasing p_j
- $1||\Sigma w_j C_j$ – Weighted SPT rule
  – tasks are scheduled in order of nondecreasing p_j/w_j
- $1|r_j|\Sigma C_j$, $1|r_j|\Sigma w_j C_j$ – NP hard
- $1|pmtn, r_j|\Sigma C_j$ – solvable by modified SPT
- $1|pmtn, r_j|\Sigma w_j C_j$ – NP hard
- $1|d_j^{\sim}|\Sigma C_j$,- solvable by modified SPT
- $1|d_j^{\sim}|\Sigma w_j C_j$ – NP hard
- $1|prec|\Sigma w_j C_j$ – NP hard

- $1|r_j|\sum C_j$ – NP hard

two <u>heuristic algorithms</u> based on two criteria for adding a task to an existing partial schedule

U – set of <u>already scheduled</u> tasks



<u>Earliest Completion Time (ECT) rule</u>

1) select task T_j with $\min\{C_j | T_j \in$ **T-U** $\}$

2) assign T_j to **U**

3) calculate $s_j = \max\{r_j, C_{\alpha|\mathbf{U}|}\}$ and $C_j = s_j + p_j$

<u>Earliest Starting Time (EST) rule</u>

1) select task T_j with $\min\{s_j | T_j \in$ **T-U** $\}$

2,3) ….

- $1|\tilde{d}_j|\Sigma w_j C_j$ – NP hard

<u>heuristic algorithm:</u> **Smith's Bacward rule**

begin

  $p := \Sigma p_j$

  while $\mathcal{T} \neq \emptyset$

    $\mathcal{T}_p := \{T_j | T_j \in \boldsymbol{\mathcal{T}}, \tilde{d}_j \geq p\}$ //set of tasks with less urgent deadlines

    select $T_j \in \mathcal{T}_p$ with maximal p_j/w_j //WSTP in $\mathcal{T}_p$

    schedule T_j at n-th position   //backward rule

    $n := n - 1;$

    $\mathcal{T} := \boldsymbol{\mathcal{T}} - \{T_j\};$

    $p := p - p_j;$

  endwhile

end

Smith's Bacward rule - cont

algorithm complexity O(n log n)

this heuristic is exact if:

(i) unit processing time $1|\ p_j=1, d_j^{\sim}|\ \Sigma w_j C_j$

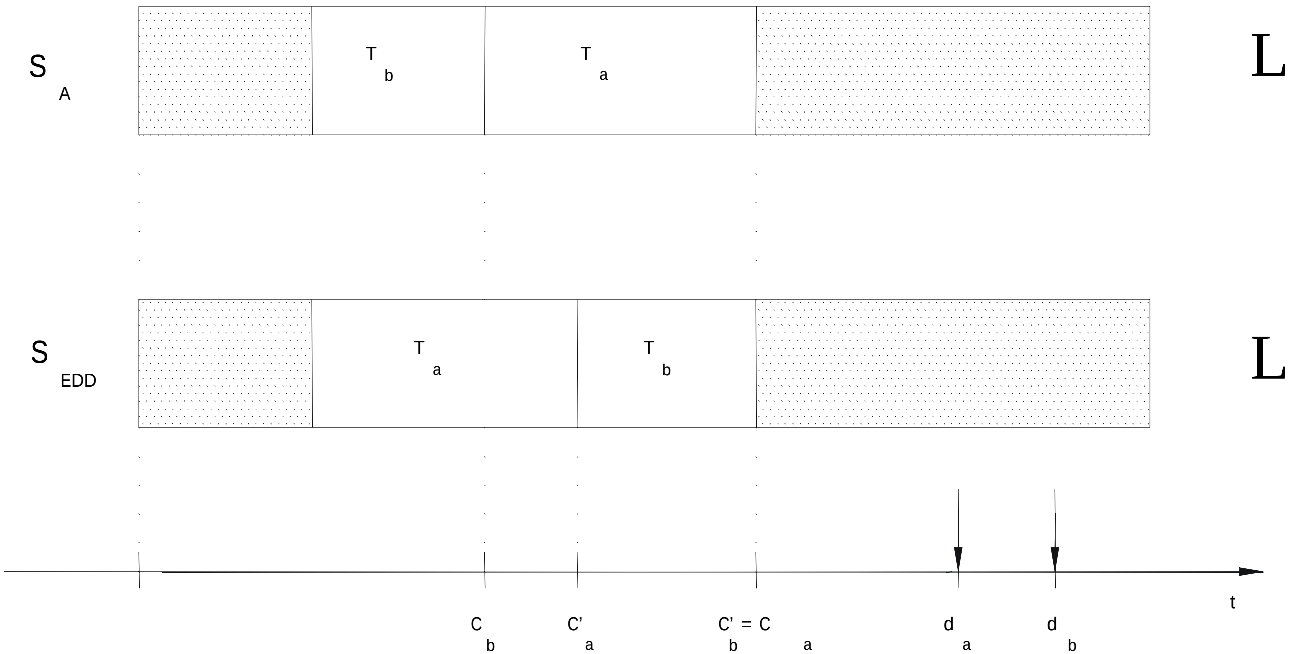
(ii) unit weights $1|d_j^{\sim}|\ \Sigma C_j$

(iii) agreeable weights, i.e. problems where $p_i \leq p_j$ implies $w_i \geq w_j$ for $i,j = 1,\ldots,n$

- $1|prec|\ \Sigma w_j C_j$ – NP hard – formulation of 0/1 programming

L_{max}

- $1||L_{max}$ – Earliest Due Date (EDD) first - Jackson
  - tasks are scheduled in order of nondecreasing due dates
  - optimality can be proven by a simple interchange:
    - Let $\boldsymbol{S_A}$ be a schedule produced by algorithm $\boldsymbol{A}$
    - If $\boldsymbol{A}$ is different than EDD, then there exist two tasks T_a and T_b with $d_a \leq d_b$, such that T_b immediately precedes T_a in $\boldsymbol{S_A}$
    - Interchanging the position of T_a and T_b cannot increase L_{max} .
    - By finite number of transpositions $\boldsymbol{S_A}$ is changed to $\boldsymbol{S_{EDD}}$.



$L_{max} = C_a - d_a$

$L'_{max} = \max \{L'_a, L'_b\}$

…. 2 cases

1. if $L'_a \geq L'_b$ then $L'_{max} = C'_a - d_a < C_a - d_a$
2. if $L'_a \leq L'_b$ then $L'_{max} = C'_b - d_b < C_a - d_a$

in both cases: $L'_{max} < L_{max}$

L_{max}

- $1|r_j|\ L_{max}$ – NP hard
- $1|r_j,\ p_j=1|\ L_{max}$ – can be solved by modified EDD
- $1|pmtn,\ r_j|\ L_{max}$ – modification of EDD by Horn
- $1|pmtn,\ r_j\ ,\ d_j=d_j^{\sim}|\ L_{max}$ – the same Horn's algorithm using EDF
- $1|pmtn,\ prec,\ r_{j,},\ d_j=d_j^{\sim}|\ L_{max}$ – transformation to independent task set and then EDF

- $1|r_j, p_j=1|\ L_{max}$ – can be solved by **modified EDD**

begin

t:=0;

while $\mathsf{T} \neq \emptyset$

$t:=\max\{t, \min_{T_j \in \mathbf{T}}\{r_j\}\}$ //shift time if no task is executable

$\mathsf{T}_r := \{T_j | T_j \in \mathbf{T}, r_j \leq t\}$ //set of execut. tasks

select $T_j \in \mathsf{T}_r$ with minimal d_j //EDD in T_r

schedule T_j at at instant t

$\mathsf{T} := \mathbf{T} - \{T_j\}$;

$t := t + 1$;

endwhile

end

- $1|pmtn, r_j|\ L_{max}$ – modification of EDD by Horn

Theorem: given a set of *n* independent tasks with arbitrary release times, any algorithm that <u>at any instant</u> executes the task with <u>earliest absolute due date</u> among all the ready tasks is optimal with respect to minimizing L_{max}

When we assume $d_j^{\sim}=d_j$ then the same applies for EDF (Earliest Deadline First) since it optimizes both:

  - **schedulability** – if there exists a feasible schedule ($L_{max} \leq 0$) for given instance, then EDF is able to find it
  - $\mathbf{L_{max}}$ - EDF minimizes L_{max}

EDF optimality

- Let S_A be a schedule produced by algorithm A and S_{EDF} by EDF.
- Without loss of generality S_A can be divided into unit time slices.
- Let $i(t)$ is id of task executing slice t
- Let $j(t)$ is id of ready task with earliest deadline at time t
- If $S_A \neq S_{EDF}$ then there is slice t such that $i(t) \neq j(t)$
- Interchanging position of $i(t)$ and $j(t)$ cannot increase maximum lateness.
- If schedule S_A starts at time t=0 and D is the latest deadline then S_{EDF} is obtained from S_A by at most D transpositions.

Figure Buttazzo page 58

EDF preserves schedulability

- …..pravdepodobne lze nahradit uvahou,ze schedulability odpovida splneni podminky $L_{max} \leq 0$, (neboli kdyz nase optimalizace nalezne reseni s $L_{max} > 0$, tak je zrejme ze to neni "schedulable" jelikoz neexistuje zadny rozvrh s mensim L_{max} …obracene je to trivialni – pokud nalezneme rozvrh s $L_{max} \leq 0$ pak je to rozvrhnutelne)
- Let $\tau(t)$ is the time ($\tau(t) > t$) at which next slice of task $j(t)$ begins its execution in current schedule
- At any instant each slice of S_A can be:

either <u>brought forward</u> – schedulability is obviously preserved

or <u>postponed</u> - if slice of T_i is postponed at $\tau(t)$ and S_A is schedulable then it must be $(\tau(t)+1) \leq d^{\sim}(t)$ being $d^{\sim}(t)$ the earliest deadline. Since $d^{\sim}(t) \leq d_i^{\sim}$ for any unexecuted T_i then we have $(\tau(t)+1) \leq d_i^{\sim}$, which guarantees schedulability of the slice postponed at $\tau(t)$.

- $1|pmtn, prec, r_{j,}, d_j^{\sim}| L_{max}$ – Chetto,Silly,Bouchentouf

Basic idea is to transform a set T of dependent tasks into T * of independent tasks by modification of timing parameters:

– modification of the release times

1) For any task without predecessors set $r_j^* = r_j$
2) Select a task T_j such that its release time has not been modified but the release times of all immediate predecessors T_h have been modified. If no such task exists, exit.
3) set $r_j^* = \max\{r_j, \max\{r_h^* + p_h \mid T_h$ is immediate predec. of $T_j\}$ and skip to step 2.

– modification of deadlines

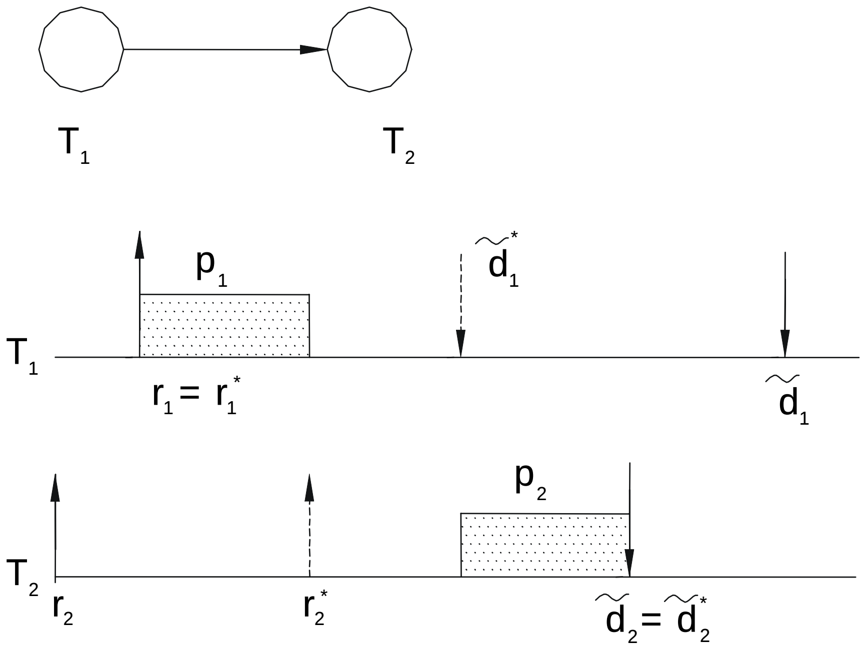
1) For any task without successors set $d_j^{\sim *} = d_j^{\sim}$
2) Select a task T_j such that its deadline has not been modified but the deadlines of all immediate successors T_k have been modified. If no such task exists, exit.
3) set $d_j^{\sim *} = \min\{d_j^{\sim}, \min\{d_k^{\sim *} - p_k \mid T_j \text{ is immediate suc. of } T_k\}$ and skip to step 2.

– EDF is executed on $\mathcal{T}^*$

Proof of optimality

– since $r_j^* \geq r_j$ and $d_j^{\sim *} \leq d_j^{\sim}$ the schedulability of $\mathcal{T}^*$ implies also schedulability of $\mathcal{T}$

– precedence constraints are not violated since due to EDF and modification of timing parameters the scheduled tasks are ordered in the same way as given by the precedence relations



- $\tilde{r}_1^{\,*} = r_1$
- $r_2^{\,*} = r_1 + p_1$
- $d_1^{\sim *} = d_2^{\sim} - p_2$
- $d_2^{\sim *} = d_2^{\sim}$